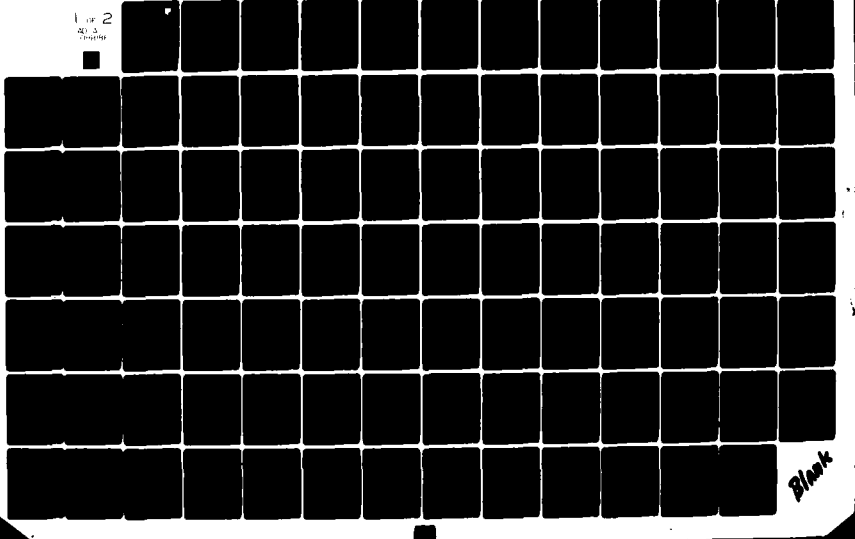
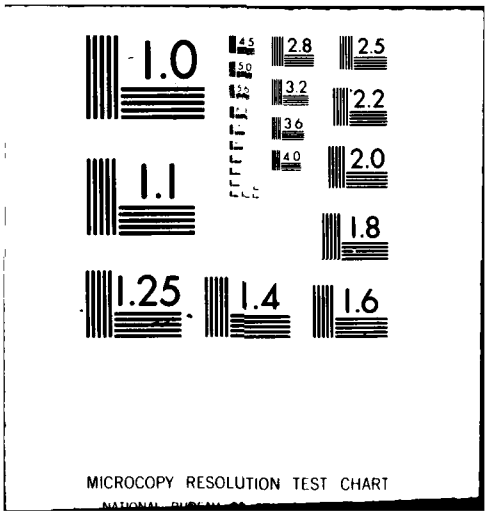
AD-A088 186 SYRACUSE UNIV N Y DEPT OF INDUSTRIAL ENGINEERING AND--ETC F/G 12/1
A TIME DEPENDENT ERROR DETECTION RATE MODEL FOR SOFTWARE PERFOR--ETC(U)
MAY 80 A L GOEL, K OKUMOTO F30602-78-C-0351

UNCLASSIFIED RADC-TR-80-179 NL

1 OF 2
AD A088186



Blank


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART

LEVEL II

12

RADC-TR-80-179
Interim Report
May 1980

A TIME DEPENDENT ERROR DETECTION RATE MODEL FOR SOFTWARE PERFORMANCE ASSESSMENT WITH APPLICATIONS

Syracuse University

Amrit L. Goel
K. Okumoto

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-80-179 has been reviewed and is approved for publication.

APPROVED: Alan N. Sukert

ALAN N. SUKERT
Project Engineer

APPROVED: Wendall C Bauman

WENDALL C. BAUMAN, Colonel, USAF
Chief, Information Sciences Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (ISIS), Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list..

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(19) REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
(18) 1. REPORT NUMBER: RADC-TR-80-179 — 2. GOVT ACCESSION NO.: AD-A088186 (9)	3. RECIPIENT'S CATALOG NUMBER
(6) 4. TITLE (and Subtitle): A TIME DEPENDENT ERROR DETECTION RATE MODEL FOR SOFTWARE PERFORMANCE ASSESSMENT WITH APPLICATIONS.	5. TYPE OF REPORT & PERIOD COVERED: Interim Technical Report, Sep 78 — Oct 79
	6. PERFORMING ORG. REPORT NUMBER: N/A
(10) 7. AUTHOR(s): Amrit L. Goel, K. Okumoto — (15)	8. CONTRACT OR GRANT NUMBER(s): F30602-78-C-0351
* 9. PERFORMING ORGANIZATION NAME AND ADDRESS: Syracuse University NY. / Department of Industrial Engineering, Syracuse NY 13210 — (16)	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 62702F, 55812015 (17) 20
11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (ISIS), Griffiss AFB NY 13441 — (11)	12. REPORT DATE: May 80 (12) 113
	13. NUMBER OF PAGES: 114
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Same	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Alan N. Sukert (ISIS)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Non-Homogeneous Poisson Process
Software Reliability
Software Error Models
Software Performance Assessment

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report presents the results of the software performance modelling task pursued under Contract No. F30602-78-C-0351 during the period September 1978 - October 1979.

The objective of this study was to develop a parsimonious model whose parameters have a physical interpretation, and which can be used to predict various quantitative measures for software performance (Cont'd)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

409154

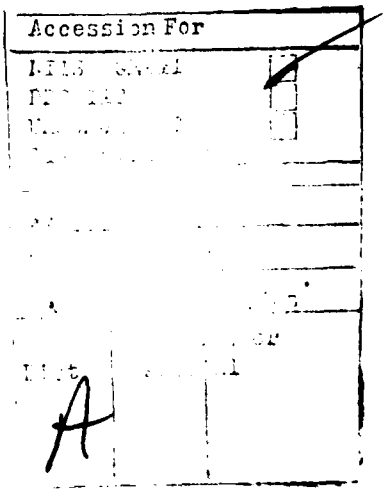
Item 20 (Cont'd)

assessment. With this objective, the behavior of the software failure counting process ($N(t)$, $t \geq 0$) has been studied. It is shown that $N(t)$ can be well described by a non-homogeneous Poisson process (NHPP) with a two parameter exponentially decaying error detection rate. Several measures, such as the number of failures by some prespecified time, the number of errors remaining in the system at a future time, and software reliability during a mission, have been proposed in this report. Models for software performance assessment are also derived.

Two methods are developed to estimate model parameters from either failure count data or times between failures. A goodness-of-fit test is also developed to check the adequacy of the fitted model.

Finally, actual failure data are analyzed from two DOD software systems. One is a large command and control system and the other a Naval data analysis system.

TABLE OF CONTENTS



LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION AND BACKGROUND

1.1 Introduction

Continued increase in the complexity and use of computer systems in a wide variety of applications, especially during the last decade, has necessitated a greater emphasis on the development of cost-effective and reliable software. The importance of software has been further enhanced by the fact that the ratio of software to hardware costs continues to grow as technological advances keep reducing the hardware cost. This has led to the evolution of a new discipline called software engineering (Reference 6). This discipline is still in its infancy and has been described as the practical application of scientific knowledge to produce software in a way that is cost-effective and reliable.

The performance of a software system is dependent on the tools and techniques used during its development and operation. An important performance criterion is the nature and frequency of software failures. A failure is said to occur when a fault, a specific manifestation of an error, in the program is evoked by some input data resulting in the computer program not correctly computing the required function. A software error is a conceptual, syntactical, or clerical discrepancy which results in one or more faults in the software. It should be noted that these definitions are controversial and not uniformly accepted. To be consistent with the existing literature, in this report the terms error and failure are used interchangeably, except where indicated otherwise.

Several empirical studies of software failure phenomena have been undertaken in recent years with the objective of improving software performance. Such studies can be classified into one (or both) of two categories. In the first category the emphasis is on the analysis of software error data collected from small or large projects (References 17, 22, 43, 56, 67, 70), during development and/or operational phases. Studies in the second category are primarily aimed at the development of analytical models (References 7, 14, 24, 25, 26, 28, 31, 36, 46, 58, 60, 61, 66, 69, 71).

A number of software reliability models have been proposed and investigated during the last seven years to describe the stochastic behavior of the software failure process and to estimate the number of software errors remaining in the system. We classify these models into two major categories. The first one emphasizes the stochastic nature of software failures, while the second approach uses combinatorial analysis to provide measures of software reliability.

The basis of the first approach is the reliability theory developed for hardware systems. Since the error detection rate changes during the software development cycle, the models have been modified to incorporate this feature of the software failure phenomenon. The time between failures is usually assumed to be exponential with a parameter that changes with the number of remaining errors in this class of models (References 26, 31, 46, 58, 61). Some work has been done using a Bayesian approach (References 23, 36) in which the time to next failure is taken to be dependent on

the previous failure history. Software reliability has also been modelled based on the number of errors found during the debugging phase (Reference 34). Also of interest in this category are the models for software availability (References 49, 50, 68).

As mentioned above, in the second approach software reliability is measured using combinatorial analysis which includes capture-recapture sampling (or error seeding) models and input data domain models. The objective of capture-recapture sampling is to determine the number of remaining errors in a computer program by introducing (or seeding) errors and then using classical statistical techniques for estimation. Computer systems are initialized and exercised by a controlled input data set and the reliability of the program is estimated by running the program for all such possible input data sets. This is the input data domain approach which has serious limitations from a practical viewpoint.

A classification of the above models is shown in Figure 1.1. Each of these categories is described in more detail in Appendix A.

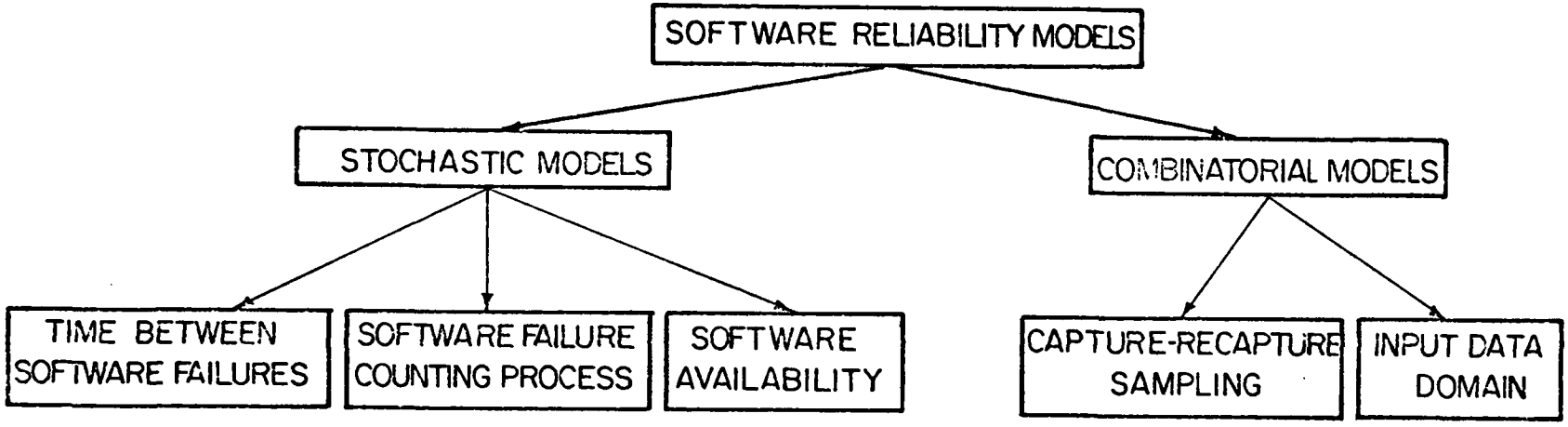


FIGURE I.I CLASSIFICATION OF SOFTWARE RELIABILITY MODELS

1.2 Purpose and Outline of the Report

In this report our objective is to develop a parsimonious model whose parameters have a physical interpretation, and which can be used to predict various quantitative measures for software performance assessment. Also of interest is the applicability of the model over a broad class of projects. Further, it should be possible to estimate the parameters of the model from available failure data which could be given as either the number of failures in specified time intervals or as times between software failures.

With this objective, we develop and investigate a non-homogeneous Poisson process (NHPP) (Reference 9) model with a time dependent error detection rate for the software failure phenomenon. By studying the behavior of the counting process, $\{N(t), t \geq 0\}$, the number of failures by time t, it is shown in Section 2 that $N(t)$ can be well described by a non-homogeneous Poisson process (NHPP) with a two parameter exponentially decaying error detection rate.

NHPP has been used by many researchers to describe random phenomena in various applications (References 13, 15, 16). Some such applications are the occurrences of coal mining disasters (Reference 39); equipment failures (References 16, 32, 51); transactions in a data-base system (Reference 33), and software error counts over a series of time intervals (Reference 60). Various forms of the intensity function for the NHPP used in actual applications are the exponential polynomial rate function (Reference 33), a log-linear rate function (Reference 11) and a Weibull rate function (References 13, 15, 44).

Several measures for software performance assessment, such as the number of errors remaining in the system, distribution of time to next failure, and software reliability are proposed in Section 3. Based on the NHPP model, expressions are then derived for obtaining the estimates and confidence limits for these performance measures.

Two methods are described in Section 4 for estimating the parameters of the model from available failure data. The first one is for the case when data is given in the form of number of failures in given time intervals. The time intervals can be of equal or unequal lengths. The second method is used when times between software failures are given.

In Section 5, a method for testing the goodness-of-fit is developed based on the Kolmogorov-Smirnov test. Expressions for performing this test are also derived.

A general methodology for analyzing software failure data is presented in Section 6. It gives a step by step procedure of analysis, starting from raw data to the computation of useable performance measures. This methodology is employed in Sections 7 and 8 to give a detailed analysis of failure data from two software systems. The first one (Section 7) is a large command and control system while the second (Section 8) is a relatively small Naval Tactical Data System (NTDS). A comparison of NTDS data analysis using the NHPP and the De-Eutrophication process models is presented in Section 9. Some concluding remarks, limitations and advantages of the NHPP model are summarized in Section 10.

2. MODEL DEVELOPMENT

A software system in use is subject to failures caused by errors present in the system. The errors are encountered when a sequence of instructions is executed which, in turn, depends on the input data set. In this section we develop a model to describe this failure occurrence phenomenon.

2.1 Deterministic Analysis of Software Failure Process

It is useful to first make a simpler analysis by ignoring the statistical fluctuations in the number of software failures before analyzing the failure phenomenon as a stochastic process (Reference 12). Let $n(t)$ denote the cumulative number of software failures detected by time t. Assume that $n(t)$ is large enough so that it can be expressed as a continuous function of t. Since the number of errors in a system is a finite value, $n(t)$ is a bounded non-decreasing function of t with

$$n(0) = 0 \quad \text{and} \quad n(\infty) = a . \tag{2.1}$$

For purposes of modeling we assume that the usage of the system is basically similar over time. Then the number of failures in $(t, t+\Delta t)$ is proportional to the number of undetected errors at t, i.e.,

$$n(t+\Delta t) - n(t) = b\{a-n(t)\}\Delta t , \tag{2.2}$$

where b is a proportionality constant.

A graphical representation of the above description is provided in Figure 2.1.

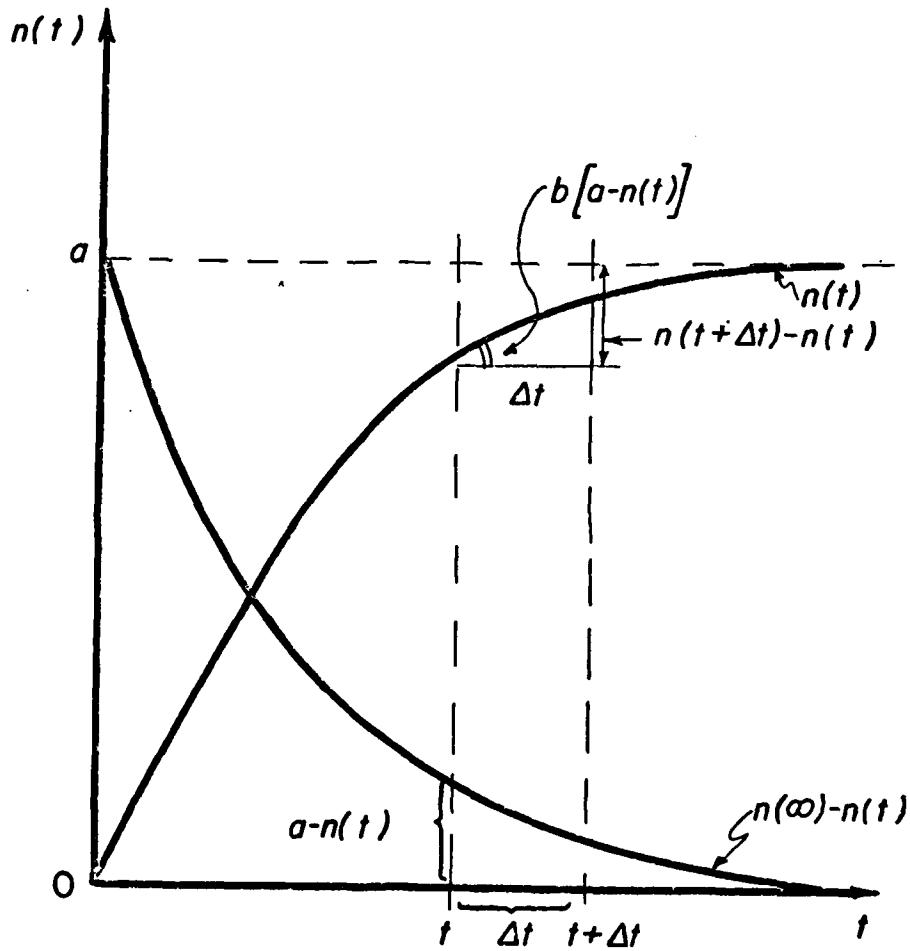


FIGURE 2.1 A GRAPHICAL REPRESENTATION OF THE DETERMINISTIC MODEL FOR SOFTWARE FAILURES

Now, from Equation (2.2) we get the differential equation

$$n'(t) = ab - bn'(t) . \tag{2.3}$$

Taking the Laplace transform (References 1, 10) of Equation (2.3) under the conditions of Equation (2.1), we have

$$s\tilde{n}(s) = ab - b\tilde{n}(s) ,$$

or

$$\tilde{n}(s) = \frac{ab}{s+b} , \tag{2.4}$$

where

$$\tilde{n}(s) = \int_0^\infty e^{-st} \cdot dn(t) . \tag{2.5}$$

The solution of Equation (2.3) is thus obtained by inverting Equation (2.4) and is given by

$$n(t) = a(1 - e^{-bt}) . \tag{2.6}$$

Under the assumptions discussed above, Equation (2.6) is the deterministic model of the software failure process. For given a and b, we can easily compute the number of failures to be encountered by some time t so that the failure phenomenon can be described with certainty. It should be noted, however, that the actual failure phenomenon is not deterministic.

2.2 Stochastic Analysis of Software Failure Process

In an actual usage the software system is subjected to random inputs causing the failures to occur at random times, i.e., the failure phenomenon is stochastic (non-deterministic). Therefore, a realistic description of the failure process must incorporate this randomness.

Let $(N(t), t \geq 0)$ be a counting process (References 52, 54, 62) representing the cumulative number of failures by time t. (Note that $N(t)$ is a random variable while $n(t)$ above was taken to be deterministic.) Assuming that each failure is caused by one error, $N(t)$ also represents the cumulative number of errors detected by time t. It should be pointed out that a detected error may not be removed and as a result may cause additional failure(s) at a later stage. For the $N(t)$ process, such recurrences are counted as new events.

Let $m(t)$ be the mean value function of the $N(t)$ process, i.e.,

$$m(t) \equiv E[N(t)] . \tag{2.7}$$

Since $m(t)$ represents the expected number of software failures or detected errors by time t, it is a non-decreasing function of t. If we assume that there will be a finite number of errors to be detected in an infinite amount of time, $m(t)$ has the following boundary conditions:

$$m(t) = \begin{cases} 0, & t = 0 \\ a, & t = \infty \end{cases} \tag{2.8}$$

where $a < \infty$ and represents the expected number of software errors

to be eventually detected. Furthermore, it is assumed that for small Δt the expected number of software failures during $(t,t+\Delta t)$ is proportional to the expected number of software errors undetected by time t, i.e.,

$$m(t+\Delta t) - m(t) = b\{a-m(t)\}\Delta t \tag{2.9}$$

where b is a constant of proportionality. Solving the differential equation obtained from Equation (2.9) under the boundary conditions of Equation (2.8), we get

$$m(t) = a(1 - e^{-bt}) . \tag{2.10}$$

This equation specifies the mean value function for the underlying software failure counting process $N(t)$. The intensity function, obtained by taking the derivative of $m(t)$, represents the error detection rate at time t and is given by

$$\lambda(t) \equiv m'(t) = abe^{-bt} . \tag{2.11}$$

We now study the probabilistic behavior of the $N(t)$ process by using $m(t)$ and $\lambda(t)$. Since there are no failures at $t=0$, we have $N(0)=0$. It is also reasonable to assume that the number of software failures during non-overlapping time intervals are independent. In other words, for any finite collection of times $t_1 < t_2 < \ldots < t_n$, the n random variables $N(t_1)$, $\{N(t_2)-N(t_1)\}$,..., $\{N(t_n)-N(t_{n-1})\}$ are statistically independent. This implies that the counting process $\{N(t), t \geq 0\}$ has independent increments.

We assign the probabilities on the increments of the N(t) process as follows.

$$N(t+\Delta t) - N(t) = \begin{cases} 0 \text{ with probability } 1-\lambda(t)\Delta t+o(\Delta t) \\ 1 \text{ with probability } \lambda(t)\Delta t+o(\Delta t) \\ 2 \text{ with probability } o(\Delta t) \\ \vdots \qquad\qquad\qquad\qquad \vdots \end{cases} \tag{2.12}$$

where

$$\frac{o(\Delta t)}{\Delta t} \to 0 \quad \text{as} \quad \Delta t \to 0 .$$

The underlying N(t) process satisfying conditions of Equation (2.12) is now a NHPP with mean value function m(t) and intensity function $\lambda(t)$ as given in Equations (2.10) and (2.11), respectively (References 18, 19). Hence the distribution of N(t) is given by

$$P\{N(t)=y\} = \frac{\{m(t)\}^y}{y!} e^{-m(t)}, \qquad y = 0,1,2,\ldots . \tag{2.13}$$

Under the assumptions discussed above, the stochastic behavior of the software failure phenomenon can be completely described by Equation (2.13). It should be pointed out that Equation (2.9) implies that the ratio

$$\frac{\text{Number of errors detected during } (t,t+\Delta t)}{(\text{Number of errors undetected by } t)\Delta t} = b \tag{2.14}$$

is constant at any time t. Therefore, b can be interpreted as the error detection rate per error.

Equations (2.10) and (2.13) constitute the basic software failure model under study in this report.

3. MODELS FOR SOFTWARE PERFORMANCE ASSESSMENT

The model developed in Section 2 is a description of the failure phenomenon. In order to use this model to predict software performance, we generally need expressions for quantitative measures such as the number of failures by some prespecified time, the number of errors remaining in the software at a future time, and software reliability during a mission. In this section we develop models that can be employed to estimate such quantities.

3.1 Number of Software Errors Detected by t

For given a and b the distribution of N(t), the cumulative number of software failures detected by time t, is obtained from Equations (2.10) and (2.13) as

$$P\{N(t)=y\} = \frac{\{a(1-e^{-bt})\}^y}{y!} e^{-a(1-e^{-bt})}, \quad y = 0,1,2,... \qquad (3.1)$$

In other words, N(t) has a Poisson distribution with mean

$$m(t) \equiv E[N(t)] = a(1-e^{-bt}). \qquad (3.2)$$

Note that

$$P\{N(\infty)=y\} = \frac{a^y}{y!} e^{-a}, \quad y = 0,1,2,.... \qquad (3.3)$$

I.e., the distribution of N(∞), the total number of failures encountered or errors detected if the system is used indefinitely, is also a Poisson distribution with mean 'a'. This result is consistent with theoretical studies which indicate that the Poisson process is the limiting distribution of many phenomena similar to the software error occurrence phenomenon (References 41, 62).

3.2 Number of Remaining Errors and Related Results

We have been considering the number of failures encountered by time t, $N(t)$. Since many of the performance measures depend on the number of errors remaining in the system, we now consider this phenomenon.

Let $\overline{N}(t)$ be the number of errors remaining in the system at time t, i.e.,

$$\overline{N}(t) \equiv N(\infty) - N(t), \tag{3.4}$$

The expectation of $\overline{N}(t)$ is given by

$$E[\overline{N}(t)] = ae^{-bt}. \tag{3.5}$$

The conditional distribution of $\overline{N}(t)$, given $N(t) = y$, is obtained as follows:

$$P\{\overline{N}(t)=x \mid N(t)=y\} = P\{N(\infty) = y+x\}$$

$$= \frac{a^{y+x}}{(y+x)!} e^{-a} \qquad x = 0,1,2,\ldots. \tag{3.6}$$

This conditional distribution is important for deciding whether the software system under development can be released or not. The decision should be made based on the number of errors remaining in the software because this quantity plays an important role in software reliability assessment. Suppose that the decision-maker conducts an experiment and finds y software errors by time t. Then, a decision might be to

Accept if $\overline{N}(t) \leq n_0$

and

Reject if $\overline{N}(t) > n_0$

where n_0 is some specified number. For this decision rule, the probability that the software system is accepted for a given number of failures y by time t is

$$P\{\text{Accept}\} = P\{\bar{N}(t) \leq n_0 | N(t) = y\}$$

and, using Equation (3.6), becomes

$$P\{\text{Accept}\} = \sum_{i=0}^{n_0} \frac{a^{y+i}}{(y+i)!} e^{-a} . \tag{3.7}$$

The conditional expectation of $\bar{N}(t)$, given $N(t) = y$, is given by

$$\begin{aligned} E[\bar{N}(t) | N(t)=y] &= E[N(\infty)-N(t) | N(t)=y] \\ &= E[N(\infty)-y] \end{aligned}$$

$$\text{or} \quad E[\bar{N}(t) | N(t)=y] = a - y . \tag{3.8}$$

Therefore, the expected number of errors remaining in the software system at time t, given that y errors have been detected during the testing period t, is simply the expected number of failures to be encountered during $[0,\infty)$ less the number of errors detected during the period $[0,t]$.

As we can see, the parameter 'a' plays a crucial role in this study.

3.3 Software Reliability

Let a sequence of random variables $\{X_i, i = 1,2,...\}$ denote a sequence of times between software failures associated with the N(t) process. Then X_i denotes the time between the (i-1)st and the ith failures. We also define

$$S_n \equiv \sum_{i=1}^{n} X_i, \qquad n = 1,2,... \tag{3.9}$$

which represents the time to the nth failure. Let $\Phi_1(x)$ be the Cumulative Distribution Function (cdf) of X_1, i.e.,

$$\Phi_{X_1}(x) \equiv P(X_1 \leq x) . \tag{3.10}$$

Note that the event $\{X_1 > x\}$ implies that there are no failures during (0,x], i.e., the event $\{N(x)=0\}$. Then using Equation (3.1) the reliability function associated with the first failure time is given by

$$R_{X_1}(x) \equiv P(X_1 > x) = P\{N(x)=0\}$$

$$\text{or} \qquad R_{X_1}(x) = e^{-a(1-e^{-bx})} . \tag{3.11}$$

Now, the cdf of X_1 can be written as

$$\Phi_{X_1}(x) = 1 - R_{X_1}(x)$$

$$\text{or} \qquad \Phi_{X_1}(x) = 1 - e^{-a(1-e^{-bx})} . \tag{3.12}$$

The Probability Density Function (pdf) is defined as

$$\varphi_{X_1}(x) = \frac{d}{dx}\,\Phi_{X_1}(x)$$

so that

$$\varphi_{X_1}(x) = abe^{-bx}\, e^{-a(1-e^{-bx})}. \tag{3.13}$$

Next consider the conditional probability distribution, $\Phi_{X_2|X_1}(x|s)$, of $\{X_2|X_1\}$. The event $\{X_2 > x|X_1 = s\}$ implies {no failures in $(s,s+x]$}. Then the conditional reliability function of the second failure, given that the first failure occurs at time s, is given by

$$\begin{aligned} R_{X_2|X_1}(x|s) &\equiv P\{X_2 > x|X_1 = s\} \\ &= P\{\text{no failures in } (s,s+x]\} \\ &= P\{N(s+x)-N(s)=0\} \\ &= e^{-[m(s+x)-m(s)]} \\ &= e^{-a[e^{-bs}-e^{-b(s+x)}]}. \end{aligned} \tag{3.14}$$

From Equation (3.14), we obtain

$$\begin{aligned} \Phi_{X_2|X_1}(x|s) &\equiv 1 - R_{X_2|X_1}(x|s) \\ &= 1 - e^{-a[e^{-bs} - e^{-b(s+x)}]} \end{aligned} \tag{3.15}$$

and

$$\begin{aligned} \varphi_{X_2|X_1}(x|s) &\equiv \frac{d}{dx}\,\Phi_{X_2|X_1}(x|s) \\ &= abe^{-b(s+x)}\, e^{-a\{e^{-bs} - e^{-b(s+x)}\}}. \end{aligned} \tag{3.16}$$

Combining Equations (3.12) and (3.16), we get the joint density of X_1 and X_2 as

$$\varphi_{X_1,X_2}(x_1,x_2) = \varphi_{X_2|X_1}(x_2|x_1)\varphi_{X_1}(x_1)$$

$$= (abe^{-bx_1})(abe^{-b(x_1+x_2)})$$

$$x\ e^{-a(1-e^{-bx_1})}e^{-a\{e^{-bx_1}-e^{-b(x_1+x_2)}\}}$$

or $$\varphi_{X_1,X_2}(x_1,x_2) = a^2b^2e^{-bx_1}e^{-b(x_1+x_2)}e^{-a\{1-e^{-b(x_1+x_2)}\}}. \quad (3.17)$$

Making the transformation $s_1=x_1$, $s_2=x_1+x_2$, the joint density of S_1 and S_2 is

$$\varphi_{S_1,S_2}(s_1,s_2) = a^2b^2e^{-bs_1}e^{-bs_2}e^{-a(1-e^{-bs_2})}. \quad (3.18)$$

In general, it can be shown that the conditional reliability function of X_k, given $S_{k-1}=s$, is given by

$$R_{X_k|S_{k-1}}(x|s) = e^{-a\{e^{-bs}-e^{-b(s+x)}\}}. \quad (3.19)$$

3.4 Conditional Distribution of $X_k|S_{k-1}$

The conditional cdf and pdf are obtained from Equation (3.19) by recalling that $R(x) = 1 - \Phi(x)$ and $\varphi(x) = \frac{d}{dx}\Phi(x)$. Thus, we have

$$\Phi_{X_k|S_{k-1}}(x|s) = 1 - e^{-a\{e^{-bs} - e^{-b(s+x)}\}} \tag{3.20}$$

and

$$\varphi_{X_k|S_{k-1}}(x|s) = abe^{-b(s+x)}\, e^{-a\{e^{-bs} - e^{-b(s+x)}\}}, \tag{3.21}$$

respectively.

As can be seen from the above equations, the time to the next failure depends on the time when the last failure occurs. It should be noted that the distributions of times between failures are improper, i.e.,

$$\varphi_{X_k|S_{k=1}}(\infty|s) = 1 - e^{-ae^{-bs}} < 1. \tag{3.22}$$

This is due to the fact that the event {no failures in $(0,\infty]$} is allowed in our model. Hence, the expectations of these quantities do not exist. This type of behavior does not cause any theoretical problems in analysis.

3.5 Joint Density of Waiting Times

As defined above, $\{X_k, k = 1,2,...\}$ denotes the sequence of times between software failures. Then

$$S_n = \sum_{i=1}^{n} X_i, \qquad n = 1,2,...$$

is called the waiting time to the nth software failure. This quantity is quite important for estimation of parameters a and b and, hence, we obtain the distribution of $\{S_1,S_2,...,S_n\}$. The distribution is obtained by using an approach similar to that used for getting Equation (3.17). The result is summarized in the following theorem.

Theorem. The joint probability density of $S_1,S_2,...,S_n$ is given by

$$\varphi_{S_1,...,S_n}(s_1,...,s_n) = (ab)^n \cdot e^{-b\sum_{i=1}^{n} s_i} \cdot e^{-a(1-e^{-bs_n})} \qquad (3.23)$$

where $s_1,s_2,...,s_n$ denote the realizations of $S_1,S_2,...,S_n$, respectively.

The density can also be written as

$$\varphi_{S_1,...,S_n}(s_1,...,s_n) = e^{-m(s_n)} \prod_{k=1}^{n} \lambda(s_k) \qquad (3.24)$$

where $\lambda(s_k) = \frac{d}{ds_k}\{m(s_k)\}$ and $m(s_k) = a(1-e^{-bs_k})$. For a proof of this theorem, see References 11 and 15.

Equation (3.23) will be used in Section 4 to estimate a and b based on observed data $\underline{s} = (s_1,...,s_n)$.

3.6 Joint Counting Probability

The property of independent increments, along with Equations (2.8) and (2.12) of Section 2, provides a complete statistical characterization for NHPP so that the joint counting probability can be determined for any collection of times $0 < t_1 < t_2 < \ldots < t_n$. That is, with $t_0 = 0$, $y_0 = 0$,

$$P\{N(t_1) = y_1, N(t_2) = y_2, \ldots, N(t_n) = y_n\}$$

$$= \prod_{i=1}^{n} P\{N(t_i) - N(t_{i-1}) = y_i - y_{i-1}\}$$

$$= \prod_{i=1}^{n} \frac{[m(t_i) - m(t_{i-1})]^{y_i - y_{i-1}}}{(y_i - y_{i-1})!} e^{-m(t_n)} . \qquad (3.25)$$

Equation (3.25) is needed for estimating the parameters a and b for given data $\{(y_i, t_i), i = 1,2,\ldots,n\}$, as will be seen in Section 4.

4. ESTIMATION OF MODEL PARAMETERS FROM FAILURE DATA

The basic models for the failure process and performance measures were developed in Sections 2 and 3, respectively. In order to use these models for software performance assessment, the only parameters to be specified are the total expected number of errors to be detected, a, and the error detection rate per error, b. In other words, for given a and b, various useful quantities can be computed from the relevant equations in Sections 2 and 3.

In general, a and b are not known for a specific software system and are estimated from the available data generated during testing. However, that is not the only way to get a and b. One may be willing to extrapolate these values based on the data from one or more "similar" systems. Another method would be to use a Bayesian approach, whereby knowledge about a and b can be expressed as prior distributions and used for performance assessment. This approach can also be used in conjunction with available data and is specially useful when failure data are scarce or expensive to collect. Formal analytical expressions for the Bayesian approach are currently under development and will be reported in a separate technical report.

The purpose of this section is to describe methods for estimating a and b from failure data. Use of these methods is discussed in Sections 7 and 8 via failure data from operational systems. Such data are generally available as

(i) total number of failures (errors) in given time intervals; and/or as

(ii) times between failures.

Most of the available data is given in the form of number of failures in given time intervals. The data on time between failures is very rare. Both of these cases are considered below. Expressions are also derived for calculating the joint confidence regions for a and b.

4.1 Estimation When Cumulative Failures are Given

We first consider the case when data are available as cumulative number of failures in given time intervals. Suppose $y_1, y_2, \ldots, y_n$ are the cumulative number of failures detected by times $t_1, t_2, \ldots, t_n$, respectively. This can also be written as data pairs $\{(y_i, t_i), i = 1,2,\ldots,n\}$. Thus the number of failures in time interval (t_{i-1}, t_i) is $(y_i - y_{i-1})$ for $i = 1,2,3,\ldots,n$, where $t_0 = 0$ and $y_0 = 0$. We will obtain the Maximum Likelihood Estimates $\hat{a}$ and $\hat{b}$ of a and b, respectively. To do this, we first write the joint density and obtain the likelihood function, and then the log-likelihood function. Next, we take the partial derivatives of the log-likelihood function with respect to a and b and equate them to zero for maximization. The solutions of the resulting two equations are the desired values $(\hat{a}, \hat{b})$.

Now, to get the joint density, we note that in our notations $y_1, y_2, \ldots, y_n$ are the observed values of $N(t_1), N(t_2), \ldots, N(t_n)$, respectively. Hence from Equation (3.25),

$$P\{N(t_1) = y_1, N(t_2) = y_2, \ldots, N(t_n) = y_n\}$$

$$= \prod_{i=1}^{n} \frac{[m(t_i) - m(t_{i-1})]^{y_i - y_{i-1}}}{(y_i - y_{i-1})!} \cdot e^{-\{m(t_i) - m(t_{i-1})\}} \qquad (4.1)$$

where $m(t_i) = a(1 - e^{-bt_i})$.

It is well known that the likelihood function for the parameters is simply the joint density of $y_1, y_2, \ldots, y_n$, but now these values are known constants. Substituting for $m(t_i)$ in Equation (4.1), the likelihood function for (a,b) given the data $(\underline{y},\underline{t})$ is

$$L(a,b|\underline{y},\underline{t}) = \prod_{i=1}^{n} \frac{\{a(e^{-bt_{i-1}} - e^{-bt_i})\}^{y_i - y_{i-1}}}{(y_i - y_{i-1})!} e^{-a(1-e^{-bt_n})} . \quad (4.2)$$

Taking the natural logarithm of Equation (4.2) yields:

$$\ell nL(a,b|\underline{y},\underline{t}) = \sum_{i=1}^{n} (y_i - y_{i-1}) \ell na + \sum_{i=1}^{n} (y_i - y_{i-1}) \ell n(e^{-bt_{i-1}} - e^{-bt_i})$$

$$- \sum_{i=1}^{n} \ell_n (y_i - y_{i-1})! - a(1-e^{-bt_n}) . \quad (4.3)$$

As mentioned above, the maximum likelihood estimates (mle's) are those values of a and b which maximize $\ell nL(a,b|\underline{y},\underline{t})$, i.e., which satisfy (for brevity we write L to denote $L(a,b|\underline{y},\underline{t})$)

$$\frac{\partial \ell nL}{\partial a} = 0$$

and $\quad . \quad (4.4)$

$$\frac{\partial \ell nL}{\partial b} = 0$$

By taking the partial derivatives of Equation (4.3) and equating them to zero, we obtain after some simplification (recall that $y_0 = 0$),

$$a(1-e^{-bt_n}) = y_n , \quad (4.5)$$

and

$$at_n e^{-bt_n} = \sum_{i=1}^{n} \frac{(y_i - y_{i-1})(t_i e^{-bt_i} - t_{i-1} e^{-bt_{i-1}})}{e^{-bt_i} - e^{-bt_{i-1}}} . \qquad (4.6)$$

As can be easily seen, all the quantities in Equations (4.5) and (4.6) are known except a and b. Since these equations do not yield simple analytical forms, we must resort to numerical methods for their solution. The resulting values of a and b are the mle's $\hat{a}$ and $\hat{b}$, respectively.

It should be pointed out that even though the mle's are the desired values, it is often useful to study the log-likelihood surface as a function of parameters a and b. For given data, a plot of the log-likelihood surface can be obtained by solving Equation (4.3) for a grid of values of a and b. If the plot is flat, it would indicate a large variability associated with the mle's while a sharp surface is an indicator of low variability. A surface with sharp rises and falls might cause problems in numerical solution of Equations (4.5) and (4.6), while a well-behaved surface would ensure rapid convergence to the values $\hat{a}$, $\hat{b}$.

4.1.1 Variance-covariance and confidence region for (a,b)

In addition to the mle's $\hat{a}$, $\hat{b}$, we generally want to quantify the region in which the true values a, b might lie with a specified degree of confidence. This is referred to as obtaining the $100(1-\alpha)\%$ joint confidence region for (a,b). In general, it is not possible to get the exact confidence region (see Reference 20) because the true distribution of $(\hat{a},\hat{b})$ is unknown. However, mle's have a very desirable property

that they are asymptotically normally distributed, if the data size is large. In practice a sample size of approximately 20 ($n \gtrsim 20$) should be satisfactory to use this property.

Also of great usefulness is the invariance property of the mle's, i.e., a function of (a,b) can be estimated by using the mle's $\hat{a}$, $\hat{b}$ and this function will also be a mle. This will be useful for estimating $\bar{N}(t)$, $R(t)$, etc.

Formally, as indicated above, the mle's are normally distributed for large n, i.e.,

$$\begin{pmatrix}\hat{a}\\ \hat{b}\end{pmatrix} \sim N\left(\begin{pmatrix}a\\ b\end{pmatrix}, \Sigma_{cov}\right) \quad \text{as} \quad n \to \infty . \tag{4.7}$$

The variance-covariance matrix represents

$$\Sigma_{cov} = \begin{pmatrix}\mathrm{Var}(a) & \mathrm{Cov}(a,b)\\ \mathrm{Cov}(b,a) & \mathrm{Var}(b)\end{pmatrix}$$

and is given by

$$\Sigma_{cov} = \begin{pmatrix}r_{aa} & r_{ab}\\ r_{ba} & r_{bb}\end{pmatrix}^{-1} \tag{4.8}$$

where

$$r_{ij} = -E\,\frac{\partial^2 \ell nL}{\partial i \partial j} \quad , \quad i,j = a,b .$$

That is,

$$r_{aa} = - E \frac{\partial^2 \ell nL}{\partial a^2} \tag{4.9}$$

$$r_{ab} = r_{ba} = - E \frac{\partial^2 \ell nL}{\partial a \partial b} \tag{4.10}$$

$$r_{bb} = - E \frac{\partial^2 \ell nL}{\partial b^2} \tag{4.11}$$

Taking the appropriate partial derivatives of Equation (4.3) and substituting in Equations (4.9), (4.10) and (4.11), we obtain after some simplification, (recall that $E[N(t_i)] \equiv m(t_i) = a(1-e^{-bt_i})$:

$$r_{aa} = \frac{1}{a} \sum_{i=1}^{n} (e^{-bt_{i-1}} - e^{-bt_i}) , \tag{4.12}$$

$$r_{ab} = r_{ba} = t_n e^{-bt_n} , \tag{4.13}$$

and

$$r_{bb} = a[\sum_{i=1}^{n} \{ (t_{i-1}^2 + t_{i-1}) e^{-bt_{i-1}} - (t_i^2 + t_i) e^{-bt_i} - t_n^2 e^{-bt_n}] . \tag{4.14}$$

Substituting these expressions in Equation (4.8), we get the variance-covariance matrix for $(\hat{a},\hat{b})$. Thus the asymptotic distribution of $(\hat{a},\hat{b})$ is completely specified if (a,b) are known. However, (a,b) are of course not known. Therefore, we use the estimates $(\hat{a},\hat{b})$ for (a,b) in Equations (4.7), (4.12), (4.13) and (4.14) to get estimates of the parameters of the asymptotic bivariate normal distribution.

Now, the correlation coefficient between $\hat{a}$ and $\hat{b}$ is estimated as

$$\rho_{\hat{a},\hat{b}} = \frac{\mathrm{Cov}(\hat{a},\hat{b})}{\sqrt{\mathrm{Var}(\hat{a}) \, , \mathrm{Var}(\hat{b})}} \, , \tag{4.15}$$

where $\mathrm{Var}(\hat{a})$, $\mathrm{Var}(\hat{b})$, $\mathrm{Cov}(\hat{a},\hat{b})$ are obtained from Equations (4.8) to (4.11).

Finally, to obtain the $100(1-\alpha)\%$ confidence regions for a and b we use the following approximation (Reference 55)

$$\ell nL(\hat{a},\hat{b}|\underline{y},\underline{t}) - \ell nL(a,b|\underline{y},\underline{t}) = \frac{1}{2} \chi^2_{2;\alpha}$$

or
$$\ell nL(a,b|\underline{y},\underline{t}) = \ell nL(\hat{a},\hat{b}|y,t) - \frac{1}{2} \chi^2_{2;\alpha} \tag{4.16}$$

where $\ell nL(\hat{a},\hat{b}|\underline{y},\underline{t})$ represents the value of the log-likelihood function at $a = \hat{a}$ and $b = \hat{b}$.

Substituting Equation (4.3) in Equation (4.16) we get

$$\sum_{i=1}^{n} (y_i - y_{i-1}) \ell n a + \sum_{i=1}^{n} (y_i - y_{i-1}) \ell n (e^{-bt_i} - e^{-bt_{i-1}})$$

$$- \sum_{i=1}^{n} \ell n (y_i - y_{i-1})! - a(1-e^{-bt_n}) = C \, , \tag{4.17}$$

where

$$C = \ell nL(\hat{a},\hat{b}|\underline{y},\underline{t}) - \frac{1}{2} \chi^2_{2;\alpha} \, . \tag{4.18}$$

Equation (4.17) defines a contour of the $100(1-\alpha)\%$ confidence region. For given data, $\hat{a}$, $\hat{b}$, and α, Equation (4.17) can be solved for those values of a and b which satisfy it. (For computational purposes, it is easier to take values of $a \, (\geq \hat{a})$ and solve for the corresponding values of b.)

4.2 Estimation When Times Between Failures Are Given

Now we consider the case when data is available in the form of times between individual failures. As mentioned earlier, such data is not common and is rarely available.

Recall that $X_1, X_2, \ldots, X_n$ denote the times between failures and $S_n = \sum_{i=1}^{n} X_i$. Then the data is in the form $\underline{x} = (x_1, x_2, \ldots, x_n)$ and $s_n = \sum_{i=1}^{n} x_i$. The distribution of times between failures was discussed in Section 3.5 and is obtained from Equations (3.23) and (3.24), as

$$\varphi_{S_1,\ldots,S_n}(s_1,\ldots,s_n) = (ab)^n e^{-b\sum_{i=1}^{n} s_i} \cdot e^{-a(1-e^{-bs_n})}.$$

The likelihood function for a, b, given $\underline{s}$, is the same as above and can be written as

$$L(a,b|\underline{s}) = (ab)^n \cdot e^{-b\sum_{i=1}^{n} s_i} \cdot e^{-a(1-e^{-bs_n})}. \tag{4.19}$$

Then the log (natural) likelihood is

$$\ell nL(a,b|\underline{s}) = n\ell na + n\ell nb - b\sum_{i=1}^{n} s_i - a(1-e^{-bs_n}). \tag{4.20}$$

To get the maximum likelihood estimates $\hat{a}$, $\hat{b}$, we take the partial derivatives of Equation (4.20) and equate them to zero, i.e.,

$$\frac{\partial \ell nL}{\partial a} = 0 \tag{4.21}$$

and

$$\frac{\partial \ell nL}{\partial b} = 0. \tag{4.22}$$

These equations yield

$$\frac{n}{a} = 1 - e^{-bs_n} \tag{4.23}$$

and

$$\frac{n}{b} = as_n \cdot e^{-bs_n} - b \sum_{i=1}^{n} s_i \,. \tag{4.24}$$

As in the first case, these equations do not yield simple analytical solutions and have to be solved numerically. The solutions of Equations (4.23) and (4.24) are the mle's $\hat{a}$ and $\hat{b}$.

Regarding the asymptotic distribution of $(\hat{a},\hat{b})$, recall that (see Section 3.4), the joint density of $S_1,\ldots,S_n$ is improper. Therefore, the asymptotic properties of mle's do not hold in this case.

To obtain the $100(1-\alpha)\%$ confidence regions for (a,b) we use the same approximation as was used in Section 4.1, viz.

$$\ell nL(\hat{a},\hat{b}|\underline{s}) - \ell nL(a,b|\underline{s}) = \frac{1}{2}\chi^2_{2;\alpha} \,. \tag{4.25}$$

From Equations (4.20) and (4.25), a contour of the $100(1-\alpha)\%$ confidence region is obtained as

$$n\ell na + n\ell nb - b\sum_{i=1}^{n} s_i - a(1-e^{-bs_n}) = C \,, \tag{4.26}$$

where

$$C = \ell nL(\hat{a},\hat{b}|\underline{s}) - \frac{1}{2}\chi^2_{2;\alpha} \,. \tag{4.27}$$

As before, Equation (4.26) can be solved for given $\underline{s}$ $\hat{a}$, $\hat{b}$, and α to get the desired contours.

5. GOODNESS-OF-FIT TEST

A non-homogeneous Poisson process model was proposed in Section 2 to describe the software failure phenomenon. The mean value function of this model was given in Equation (2.10). In this section we describe the Kolmogorov-Smirnov goodness-of-fit test (K-S Test) to check whether this model provides a good fit to a given set of failure data.

Consider the case when the data are given as a sequence of software failure times $\underline{s} = (s_1, s_2, \ldots, s_n)$. We want to test whether the events $\underline{s}$ are generated from a NHPP. Suppose that $0 \le S_1 \le S_2 \le \ldots \le S_n$ are the random times at which the first n events occur in a NHPP with unknown mean value function $m(t)$. We wish to test the simple hypothesis

$$H_0 : m(t) = m_0(t) \quad \text{for} \quad t \ge 0,$$

versus $$H_1 : m(t) \neq m_0(t) \quad \text{for} \quad t \ge 0.$$

Writing $m_0(t) = a_0(1-e^{-b_0 t})$, the hypothesis H_0 can be written as

$$H_0 : m(t) = a_0(1-e^{-b_0 t}) \quad \text{for} \quad t \ge 0. \tag{5.1}$$

For testing purposes we need the joint conditional distribution of the failure times. The following theorem is useful in deriving this distribution.

Theorem 5.1. Given that $N(t) = n$, the n failure times $0 \le S_1 \le S_2 \le \ldots \le S_n$ in the interval $[0,t]$ are random variables

whose joint conditional distribution is the same as the distribution of the order statistics of a random sample of size n from the distribution $G(x) = \frac{m(x)}{m(t)}$ for $0 \leq x \leq t$.

For proof of Theorem 5.1 see Cox and Lewis (Reference 11).

Corollary 5.1. Given that $S_n = t$, the (n-1) failure times $0 \leq S_1 \leq S_2 \leq \ldots \leq S_{n-1}$ have the same joint conditional distribution as the order statistics of a random sample of size (n-1) from the distribution $G(x) = \frac{m(x)}{m(t)}$.

This Corollary easily follows from Theorem 5.1.

Using Corollary 5.1, we reduce the hypothesis of Equation (5.1) to

$$H_0 : G(x) = G_0(x) = \frac{m_0(x)}{m_0(t)} \quad \text{for} \quad 0 \leq x \leq t. \tag{5.2}$$

For our case we have

$$H_0 : G(x) = \frac{1-e^{-b_0 x}}{1-e^{-b_0 t}} \quad \text{for} \quad 0 \leq x \leq t. \tag{5.3}$$

Note that the expression in Equation (5.3) represents a truncated exponential distribution.

We now consider the Kolmogorov-Smirnov (K-S) goodness-of-fit test (References 53, 55). Given the values of a random sample of size $n-1$, $s_1, s_2, \ldots, s_{n-1}$, we define the sample cdf by $H_{n-1}(x) = k/(n-1)$, where k is the number of sample values $\leq x$. Thus $H_{n-1}(x)$ is a step function which is zero for x less than s_1, has a jump of $1/(n-1)$ at each s_k, and is 1 for x greater than or equal to s_{n-1}. That is,

$$H_{n-1}(x) = \begin{cases} 0 & , \ x < s_1 \\ k/(n-1) & , \ s_{k-1} \le x < s_k , \quad k = 2,3,\ldots,n-1 . \\ 1 & , \ x \ge s_{n-1} \end{cases} \qquad (5.4)$$

Since H_{n-1} is a step function and G is monotonically increasing and continuous, it suffices to test the absolute deviations at the sample points s_k, $k = 1,2,\ldots,n-1$, and then take the maximum of these (n-1) values. The following procedure is used for calculating the test statistic D. For each $k = 1,2,\ldots,n-1$, set

$$D_k = \max\left\{ \left| G_0(s_k) - \frac{k}{n-1} \right| , \quad \left| G_0(s_k) - \frac{k-1}{n-1} \right| \right\} .$$

Then set

$$D = \max_k \{D_k\} . \qquad (5.5)$$

If the value of D calculated in Equation (5.5) is greater than or equal to the critical value $D_{n-1;\alpha}$, we reject the null hypothesis H_0 that $S_1, S_2, \ldots, S_{n-1}$ follow $G_0(x)$; otherwise we do not reject the null hypothesis. The critical values $D_{n-1;\alpha}$ associated with the K-S test at a level of significance α are available from statistical tables (see Reference 53, p. 661).

It should be noted that if the parameters of $G_0(x)$ are estimated from the sample, the K-S test can be used but will give extremely conservative results. To achieve better results, the level of significance needs to be adjusted. One approach suggested by Allen (Reference 2) is to test at the 5% level of significance and use the critical value for the 20% level or test at the 1% level and use the critical value for 10% level. We will use this approach in our analyses in Sections 7 and 8.

Another use of the K-S test in our context is in developing confidence limits for the true cdf $G(x)$. For example, if we take a random sample of size $(n-1)$ and use it to construct the sample cdf $H_{n-1}(x)$, then we can be $100(1-\alpha)\%$ confident that the true cdf $G(x)$ does not deviate from $H_{n-1}(x)$ by more than $D_{n-1;\alpha}$. That is, the $100(1-\alpha)\%$ confidence limits for $G(x)$ are given by

$$H_{n-1}(x) - D_{n-1;\alpha} < G(x) < H_{n-1}(x) + D_{n-1;\alpha} . \qquad (5.6)$$

These limits are especially useful in the case when the parameters of $G_0(x)$ are to be estimated from the data. For this case the null hypothesis H_0 will be rejected at a level of significance α if one or more points of $G_0(x)$ fall outside the $100(1-\alpha)\%$ confidence limits given by Equation (5.6). Otherwise, it will not be rejected.

6. GENERAL METHODOLOGY FOR SOFTWARE FAILURE DATA ANALYSIS

Sections 2 through 5 were devoted to the development of models, measures, estimation techniques and a goodness-of-fit test. In this section we summarize the procedure for analyzing actual software failure data. Analyses of failure data from two typical systems are presented in Sections 7, 8 and 9.

The step by step procedure is shown in Figure 6.1 and described below.

Step 1: Study the failure data.

The model described in this report assumes that the failure data represents the data collected after the system has been integrated and the number of failures per unit time is statistically decreasing. If, however, this is not the case, the NHPP model of Section 2 may not yield satisfactory results. Furthermore, adequate amount of data must be available to get a satisfactory model. A rule of thumb would be to have at least ten data points.

Step 2: Obtain estimates $\hat{a}$ and $\hat{b}$ of parameters a and b, respectively.

Two methods are available depending upon the type of available data.

If the data is in the form of pairs (t,y), the maximum likelihood estimates are obtained by simultaneously solving Equations (4.5) and (4.6). [Section 4.1]

If the data is in the form of times between failures, the maximum likelihood estimates are obtained by simultaneously solving Equations (4.23) and (4.24). [Section 4.2]

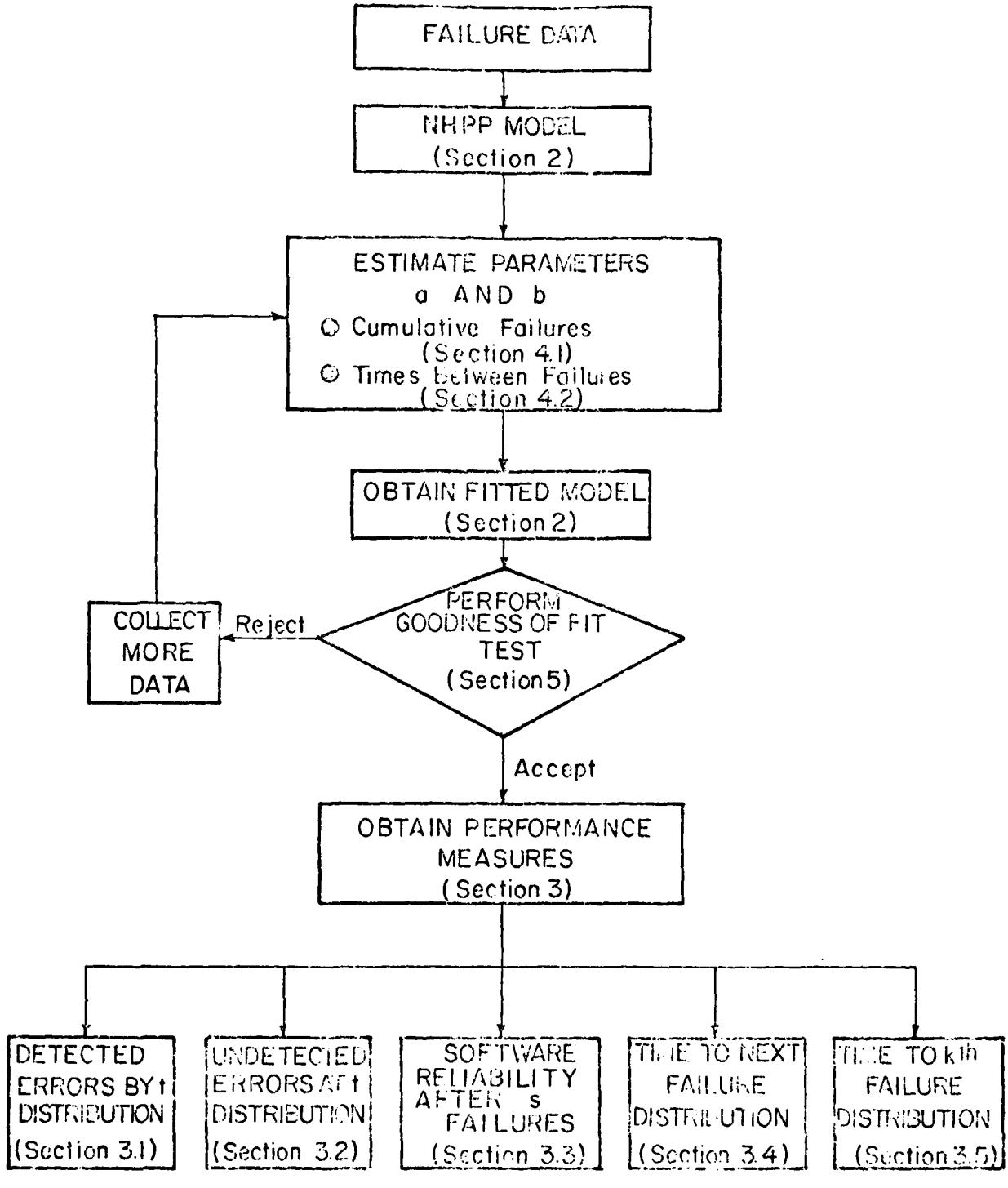


FIG. 6.1 FLOWCHART FOR SOFTWARE FAILURE DATA ANALYSIS

Step 3: Obtain the fitted model.

The fitted model is obtained by first substituting $\hat{a}$ and $\hat{b}$ from Step 2 for a and b, respectively, in Equation (2.10) to get $\hat{m}(t)$, and then substituting $\hat{m}(t)$ for $m(t)$ in Equation (2.13).

At this stage, we have a fitted model based on the available failure data.

Step 4: Perform goodness-of-fit test.

Before proceeding further, it is advisable to conduct the Kolmogorov-Smirnov goodness-of-fit test to check the model fit. This is done by following the procedure of Section 5. Specifically, the observed value of D is obtained from Equation (5.5) and compared with the critical value $D_{n;\alpha}$ for a desired significance level α. In general $\alpha = .05$ or $.10$ is quite satisfactory.

If the model fits, we can move ahead. However, if the model does not fit, we have to collect additional data or seek a better, more appropriate model. There is no easy answer to either how much more data to collect or how to look for a better model. Decisions on these issues are very much problem dependent.

Step 5: Compute confidence regions.

It is generally desirable to obtain 80%, 90%, 95% and 99% joint confidence regions for the parameters a and b using the method described in Section 4. Such regions are given by Equation (4.17) for cumulative failures data and by Equation (4.26) for the times between failures data.

Step 6: Obtain performance measures.

At this stage we can compute various quantitative measures to assess the performance of the software system. Several useful measures and expressions were given in Section 3. Equations (3.1),

(3.3), (3.5), (3.6), (3.8) and (3.19) can be used for this purpose. The specific measures to be employed will vary from one application to another. Confidence bounds can also be obtained for these measures to evaluate the degree of uncertainty in the computed values.

7. ANALYSIS OF FAILURE DATA FROM A LARGE SCALE SOFTWARE SYSTEM

The data to be analyzed in this section have been taken from a large scale project reported in Thayer et al. (1976). This project represents an initial delivery of a large command and control software package written in JOVIAL/J4, (JOVIAL is a higher order language generally used for Air Force Command and Control applications). It consists of 115,346 total source statements and 249 routines. Some other characteristics of this project are summarized in Table 7.1.

7.1 Failure Data

The failure data used for this study is taken from the Software Problem Reports (SPR's) generated during the formal testing phases of this project. The majority of software errors were detected during Validation (Jun 1-Aug 12), Acceptance (Aug 13-Aug 24), Integration (Aug 25-Oct 26), and Operational Demonstration (Oct 27-Nov 12) testing. However, operational data spanning a period of approximately nine months was also available and is used for comparison with the predicted values. The only time frame readily available from the data was the calendar day. The data also contain the mistakes by the operators and the "explanatory" errors, i.e., corrections to make a change to a comment statement or those errors for which a "fix" is not to a routine. These explanatory errors do or do not indicate the type of change. Therefore, the original data was restructured into four sets of data denoted by

TABLE 7.1

SOFTWARE PROJECT CHARACTERISTICS

Size (Total source statement)	115,346
Number of routines	249
Language	JOVIAL/J4
Formal Requirements	To function level
Co-contractor	Yes
Subcontractor	No
Operating Mode	Batch
Formal Testing	24 Weeks
Validation	10
Acceptance	2
Integration	10
Operational Demonstration	2

DS1, DS2, DS3 and DS4 (Reference 63). The description and the total number of errors detected during the formal testing phases for each data set are given in Table 7.2.

In this analysis the number of software errors detected during formal testing is counted on a weekly basis. Also, for each data set the software errors detected during the first nine weeks are eliminated due to the fact that we are interested in analyzing the software failures over the period when they are decreasing. The number of SPR's for the 15-week period for the four cases (DS1 to DS4) are given in Table 7.3.

TABLE 7.2

DESCRIPTION OF THE DATA SETS

DATA SET	DESCRIPTION	TOTAL NUMBER OF ERRORS	
		Formal Testing (24 weeks)	Operation (36 weeks)
DS1	Original Data - TT - EX1 - EX2	2191	198
DS2	Original Data - TT - EX1	2621	263
DS3	Original Data - TT	4367	540
DS4	Original Data - TT - EX2	3937	475

TT represents the mistakes by the operators.

EX1 represents the explanatory errors which do not indicate what type of change (module, documentation, compool, data base) was involved.

EX2 represents the explanatory errors which indicate type of change.

TABLE 7.3

SOFTWARE DATA SETS DS1 TO DS4

	DATA SET							
	DS1		DS2		DS3		DS4	
WEEK	# OF SPR's	CUMULATIVE	# OF SPR's	CUMULATIVE	# OF SPR's	CUMULATIVE	# OF SPR's	CUMULATIVE
1	203	203	238	238	369	369	334	334
2	136	339	179	417	292	667	255	589
3	183	522	222	639	398	1065	359	948
4	47	569	73	712	115	1180	89	1037
5	46	615	56	768	83	1263	73	1110
6	71	686	88	856	150	1413	133	1243
7	54	740	78	934	142	1555	118	1361
8	57	797	92	1026	172	1727	137	1498
9	80	877	104	1130	202	1929	178	1676
10	64	941	85	1215	168	2097	147	1823
11	27	968	53	1268	89	2186	63	1886
12	42	1010	45	1313	87	2273	84	1970
13	55	1065	59	1372	111	2384	107	2077
14	62	1127	84	1456	253	2637	231	2308
15	11	1138	27	1483	70	2707	54	2362

7.2 Estimation of Parameters

The data for this project are in the form $(t_1,y_1),(t_2,y_2),$ $\ldots,(t_{15},y_{15})$, i.e., as the number of failures in specified time intervals. Hence the estimates $\hat{a}$ and $\hat{b}$ are obtained by simultaneously solving Equations (4.5) and (4.6). Thus, by substituting the data set DS1 in Equations (4.5) and (4.6) and solving, we get

$$\hat{a} = 1348 , \qquad \hat{b} = 0.124 ,$$

and the fitted mean value function is

$$\hat{m}(t) = 1348(1 - e^{-0.124t}) , \qquad t \geq 0 .$$

This is also an estimate of the expected number of software failures detected by time t. A plot of the actual cumulative number of failures and the fitted values is given in Figure 7.1.

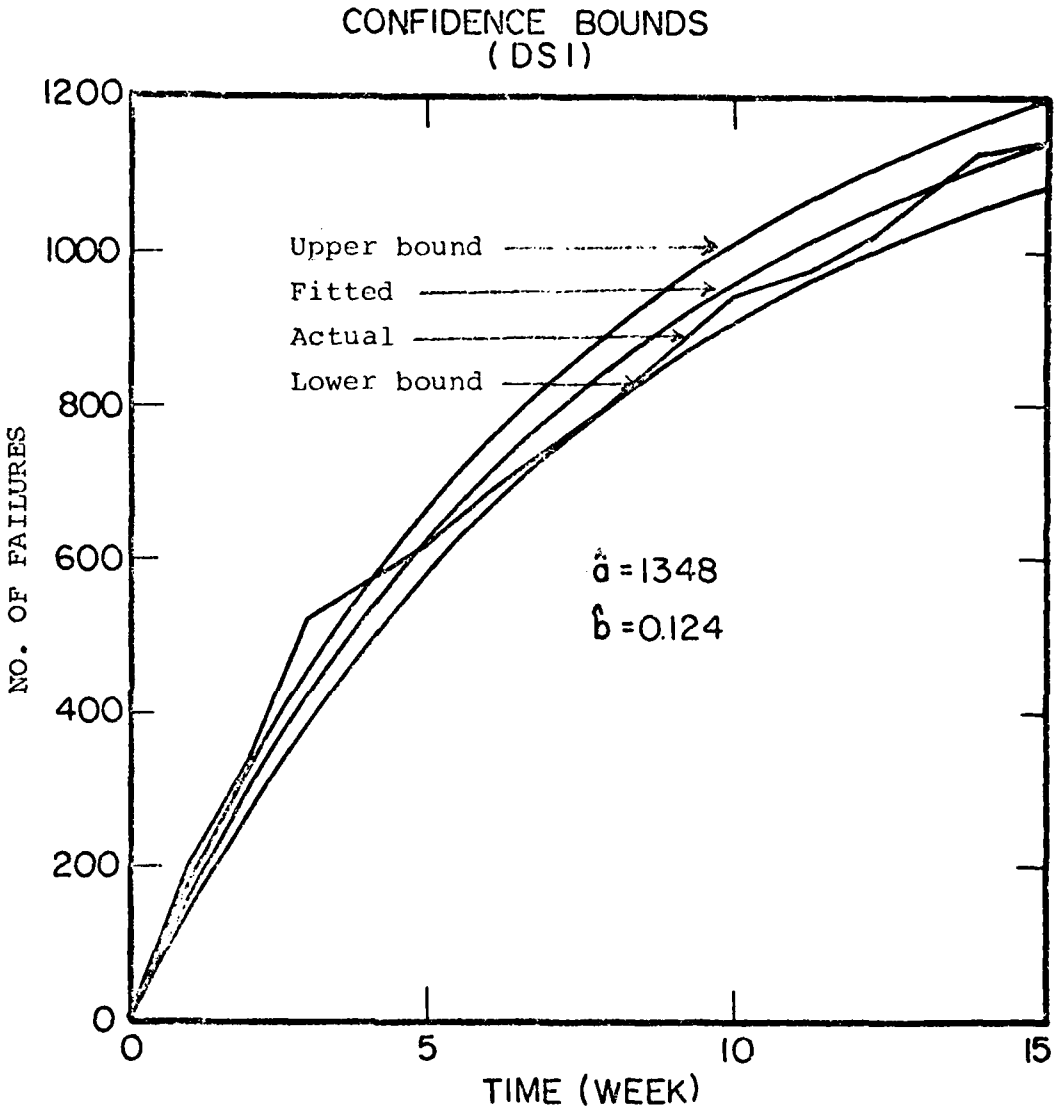


Figure 7.1 Actual and Expected Cumulative Number of Failures and 90% confidence bounds for the N(t) process for data set DS1.

7.3 Goodness-of-fit Test

The goodness-of-fit test is now conducted following the procedure discussed in Section 5. Since the sample size is 15, the null hypothesis to be tested can be written as

$$H_0 : G_0(t_i) = \frac{1-e^{-b_0 t_i}}{1-e^{-b_0(15)}} \quad \text{for} \quad i = 1,2,\ldots,15 , \qquad (7.1)$$

and the sample cdf as

$$H(x) = \begin{cases} 0 , & x < t_1 \\ y_i/y_{15} , & t_{i-1} < x < t_i , \quad i = 2,3,\ldots,15. \\ 1 , & x \geq t_{15} \end{cases} \qquad (7.2)$$

The computed values of $H(x)$ for various t_i are given in column 2 of Table 7.4.

Now we substitute $b_0 = \hat{b} = 0.124$ in Equation (7.1) and compute the value of $G_0(t_i)$ for $i = 1,2,\ldots,15$. These values are given in column 3 of Table 7.4. Columns 4 and 5 of this table are the quantities needed to find $D = \max_k \{D_k\}$ (see Equation (5.5)). From these columns we find the value of D to be 0.096 corresponding to $t_i = 9$.

To find the critical value corresponding to sample size 15 and $\alpha = .05$, we first note that the parameters had to be estimated in this case. As mentioned in Section 5, for a situation like this a suggested approach is to take $\alpha = .20$ to get good results. From the statistical tables (Reference 53, p. 661), $D_{15;.20} = 0.266$. The observed value $D = 0.096$ is less than the critical value 0.266 and hence we accept the null hypotheses of Equation (7.1). Thus

TABLE 7.4

DATA FOR KOLMOGOROV-SMIRNOV TEST
(DATA SET DS1)

t_i	$H(t_i)$	$G_0(t_i)$	$\lvert G_0(t_i)-H(t_i)\rvert$	$\lvert G_0(t_i)-H(t_{i-1})\rvert$
1	0.1784	0.1381	0.0403	0.1381
2	0.2979	0.2601	0.0378	0.0817
3	0.4587	0.3679	0.0908	0.0700
4	0.5000	0.4631	0.0369	0.0044
5	0.5404	0.5472	0.0068	0.0472
6	0.6028	0.6215	0.0187	0.0811
7	0.6503	0.6872	0.0369	0.0844
8	0.7004	0.7452	0.0448	0.0949
9	0.7707	0.7964	0.0257	0.096
10	0.8269	0.8416	0.0147	0.0709
11	0.8506	0.8816	0.031	0.0547
12	0.8875	0.9169	0.0294	0.0663
13	0.9359	0.9481	0.0122	0.0606
14	0.9903	0.9757	0.0146	0.0398
15	1.0000	1.0000	0.0000	0.0097

we conclude that at 5% level of significance the model

$$P\{N(t)=y\} = \frac{\{1348(1-e^{-0.124t})\}}{y!}\{e^{-1348(1-e^{-0.124t})}\}$$

can be considered to provide an adequate fit to data set DS1.

To further check the adequacy of fit, we compute 95% confidence bounds on $G(t_i)$. From Equation (5.6), these bounds are given by

$$H(t_i) - D_{15;.05} \leq G(t_i) < H(t_i) + D_{15;.05}.$$

From the statistical tables, $D_{15;.05} = 0.366$ and hence the 95% confidence bounds are given by $H(t_i) \pm 0.366$. A plot of these bounds and the fitted values are shown in Figure 7.2.

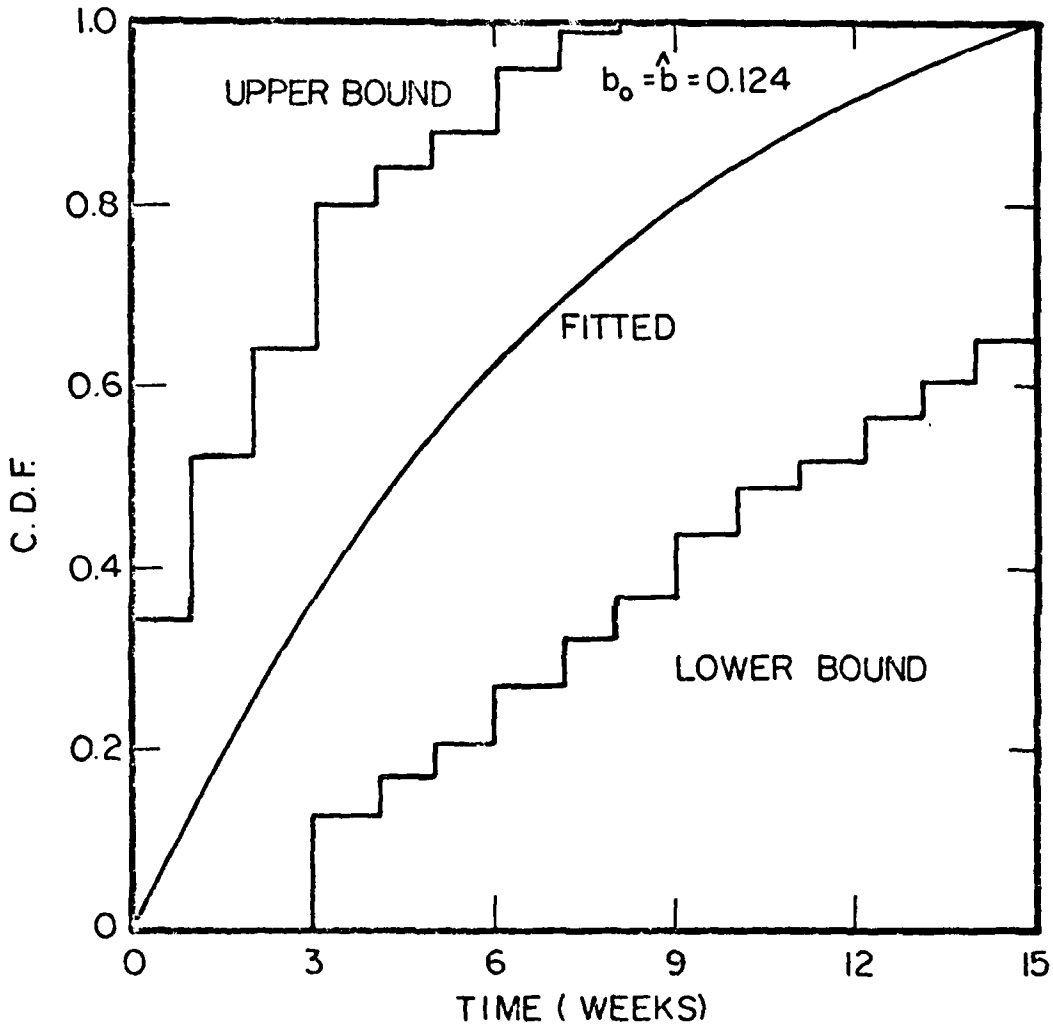


Figure 7.2 95% confidence bounds for the conditional c.d.f. $G(t_i)$ and the fitted curve for DS1 data

7.4 Confidence Regions for (a,b)

To get an appreciation of the variability in the estimated values of a and b, we now construct confidence regions for (a,b). Such regions are given by Equations (4.17) and (4.18). For $\alpha = .05$, the 95% joint confidence region will be the solution of the following equation:

$$\sum_{i=1}^{15} (y_i - y_{i-1}) \ell n 1348 + \sum_{i=1}^{15} (y_i - y_{i-1}) \ell n (e^{-.124t_i} - e^{-.124t_{i-1}})$$

$$- \sum_{i=1}^{15} \ell n (y_i - y_{i-1})! - a(1 - e^{-.124t_{15}}) = C,$$

where

$$\ell n L(\hat{a}, \hat{b} | \underline{y}, \underline{t}) = \sum_{i=1}^{15} (y_i - y_{i-1}) \ell n (1348) + \sum_{i=1}^{15} (y_i - y_{i-1}) \cdot \ell n (e^{-.124t_i} - e^{-.124t_{i-1}})$$

$$- \sum_{i=1}^{15} \ell n (y_i - y_{i-1})! - 1348(1 - e^{-.124t_{15}}),$$

and

$$C = \ell n L(\hat{a}, \hat{b} | \underline{y}, \underline{t}) - \frac{1}{2} \chi^2_{2;.05}.$$

Data $(y_1, t_1), (y_2, t_2), \ldots, (y_{15}, t_{15})$ were given in Table 7.3 and

$$\chi^2_{2;.05} = 0.103.$$

A plot of this region is shown in Figure 7.3. From this plot we see that even though the most likely values of a and b, based on the data, are $\hat{a} = 1348$, $\hat{b} = 0.124$, the true values can vary over the entire region contained in the 95% contour. Values $a = 1450$,

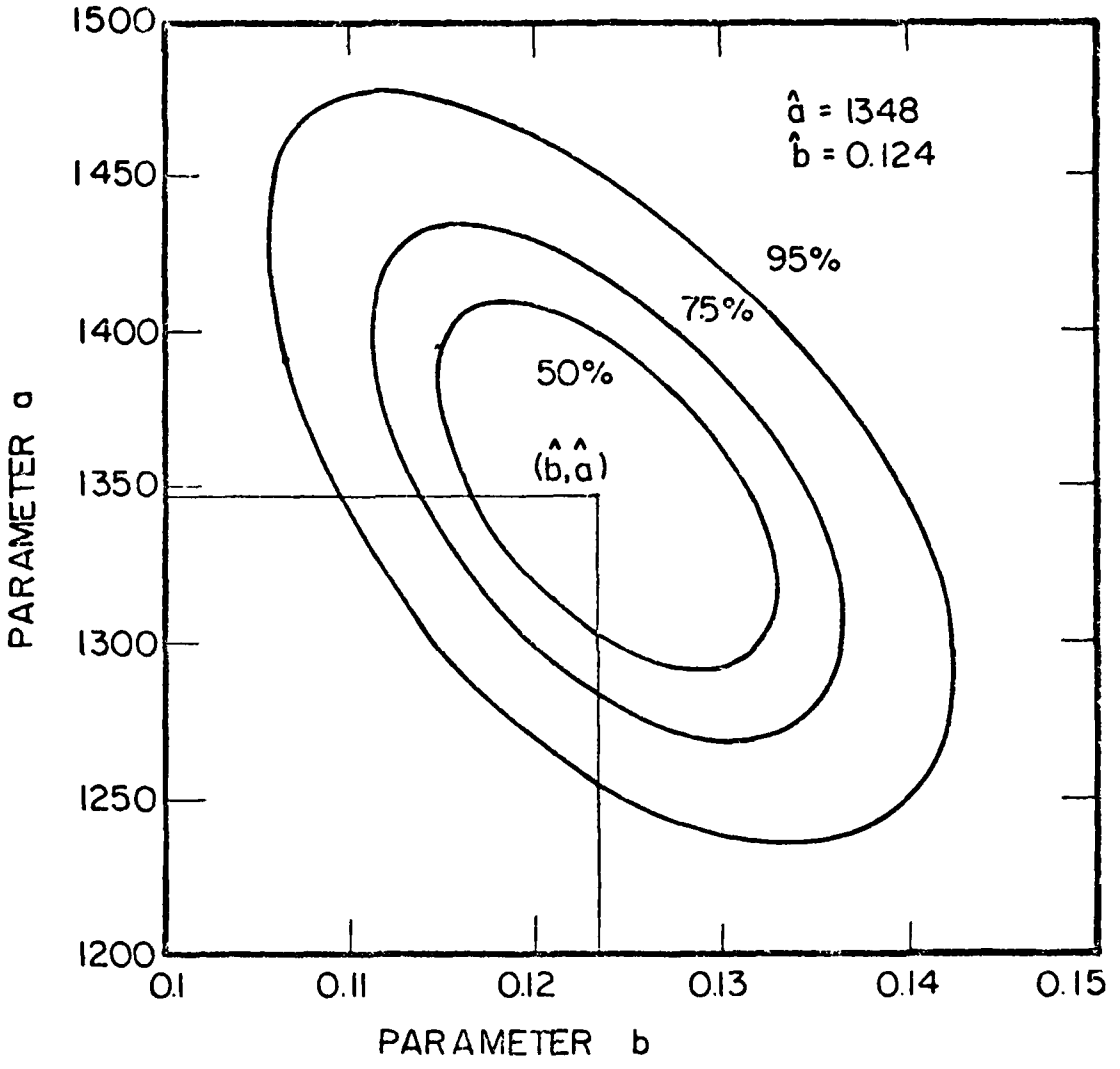


Figure 7.3 Joint confidence regions for a and b for Data Set DS1.

$b = 0.11$ will be acceptable (with 95% confidence) and so will $a = 1250$, $b = 0.14$. 50% and 75% confidence regions are also shown in Figure 7.3 and can be similarly interpreted.

7.5 Variance-Covariance Matrix

The variance-covariance matrix is useful in quantifying the variability in the estimated parameters and is obtained from Equations (4.8), (4.12), (4.13) and (4.14) by substituting $a = \hat{a} = 1348$, $b = \hat{b} = 0.124$, and the actual data values from Table 7.3. For data set DS1, we get

$$\hat{\Sigma}_{cov} = \begin{pmatrix} 2368 & -0.2071 \\ -0.2071 & 5.554 \times 10^{-5} \end{pmatrix} .$$

From this we have

$$\text{Standard Deviation } (\hat{a}) \equiv \sqrt{\text{Var}}\,(\hat{a}) = 48.66$$

$$\text{Standard Deviation } (\hat{b}) \equiv \sqrt{\text{Var}}\,(\hat{b}) = 0.00745$$

$$\text{Correlation Coefficient } (\hat{a},\hat{b}) \equiv \hat{\rho}_{\hat{a},\hat{b}}$$

$$= \frac{-0.2071}{\sqrt{(2368)(5.554 \times 10^{-5})}} = -0.571 .$$

7.6 Number of Remaining Errors

One useful quantity is the estimated number of remaining errors in the system after some time t. This value is obtained from Equation (3.5) as

$$E\{\bar{N}(t)\} = \hat{a} \cdot e^{-\hat{b}t}$$

$$\text{or} \quad E\{\bar{N}(t)\} = 1348e^{-0.124t} .$$

A plot of this quantity is shown in Figure 7.4. As expected, this value decreases with time. Also shown is a plot of the "actual" number of remaining errors which is based on the assumption that all the errors were found during 36 weeks of operation. It should be noted that this assumption is made for illustration purposes only and, in general, this may not be the case.

It would also be interesting to compute confidence bounds on $E\bar{N}(t)$. Such bounds can be easily computed as follows.

Let $f(a,b)$ denote $E\bar{N}(t)$. Then, it is well known (References 53, 55) that $100(1-\alpha)\%$ confidence bounds for $f(a,b)$ are given by

$$\{\hat{f}(a,b) \pm t_{n-2;\alpha/2}\sqrt{\hat{V}(\hat{f}(a,b))}\} . \tag{7.3}$$

where

$$\hat{V}(\hat{f}(a,b)) = \left(\frac{\partial f}{\partial a} \; \frac{\partial f}{\partial b}\right) \Sigma_{cov} \left(\begin{matrix} \frac{\partial f}{\partial a} \\ \frac{\partial f}{\partial b} \end{matrix}\right)\Bigg|_{a=\hat{a},\, b=\hat{b}} \tag{7.4}$$

and $t_{n-2;\alpha/2}$ is the upper $\alpha/2$ percentage point of the t-distribution with $(n-2)$ degrees of freedom.

The 90% confidence limits for $E\{\bar{N}(t)\}$ for data set DS1 are computed from the above equations and are plotted in Figure 7.4.

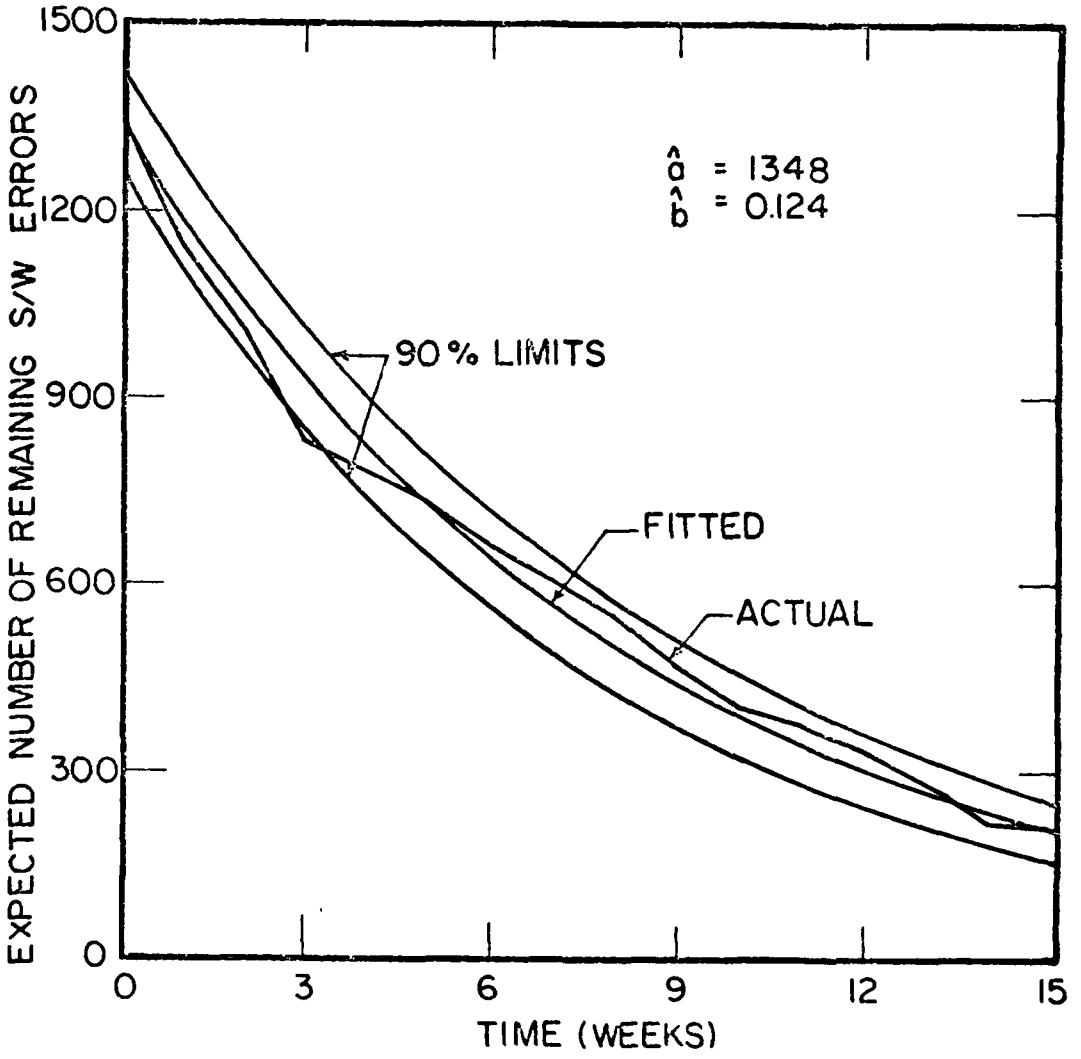


Figure 7.4 Expected number of remaining software errors and related quantities for various t (Data Set DS1)

7.7 Software Reliability

Software reliability is a commonly used performance measure to assess how reliable the system is at various times. To compute software reliability, we use Equation (3.19) and get

$$\hat{R}_{X_k|S_{k-1}}(x|s) = e^{-\hat{a}(e^{-\hat{b}s} - e^{-\hat{b}(s+x)})}.$$

This gives the reliability after time x starting from the current time s. For example, starting from $s = 15$, the reliability after 0.04 weeks, i.e., at $s+x = 15.04$, is

$$\hat{R}(0.04|s=15) = e^{-1348(e^{-(.124)15} - e^{-(.124)(15.04)})}$$

or $$\hat{R}(15.04) = 0.354.$$

To see how reliability varies with time, a plot of $\hat{R}(x|s=15)$ is shown in Figure 7.5.

To obtain confidence bounds on reliability, we use a procedure similar to the one used for getting bounds on $E\{\bar{N}(t)\}$. Let $\hat{g}(a,b)$ represent $\hat{R}(x|s=15)$. Then the confidence bounds are given by

$$\{\hat{g}(a,b) \pm t_{n-2;\alpha/2}\sqrt{\hat{V}(\hat{g}(a,b))}\}, \tag{7.5}$$

where

$$\hat{V}(\hat{g}(a,b)) = \left(\frac{\partial g}{\partial a} \; \frac{\partial g}{\partial b}\right) \Sigma_{cov} \left(\begin{array}{c} \frac{\partial g}{\partial a} \\ \\ \frac{\partial g}{\partial b} \end{array}\right)\Bigg|_{\substack{a=\hat{a} \\ b=\hat{b}}} \tag{7.6}$$

90% confidence bounds computed from these equations for the given data are shown in Figure 7.5.

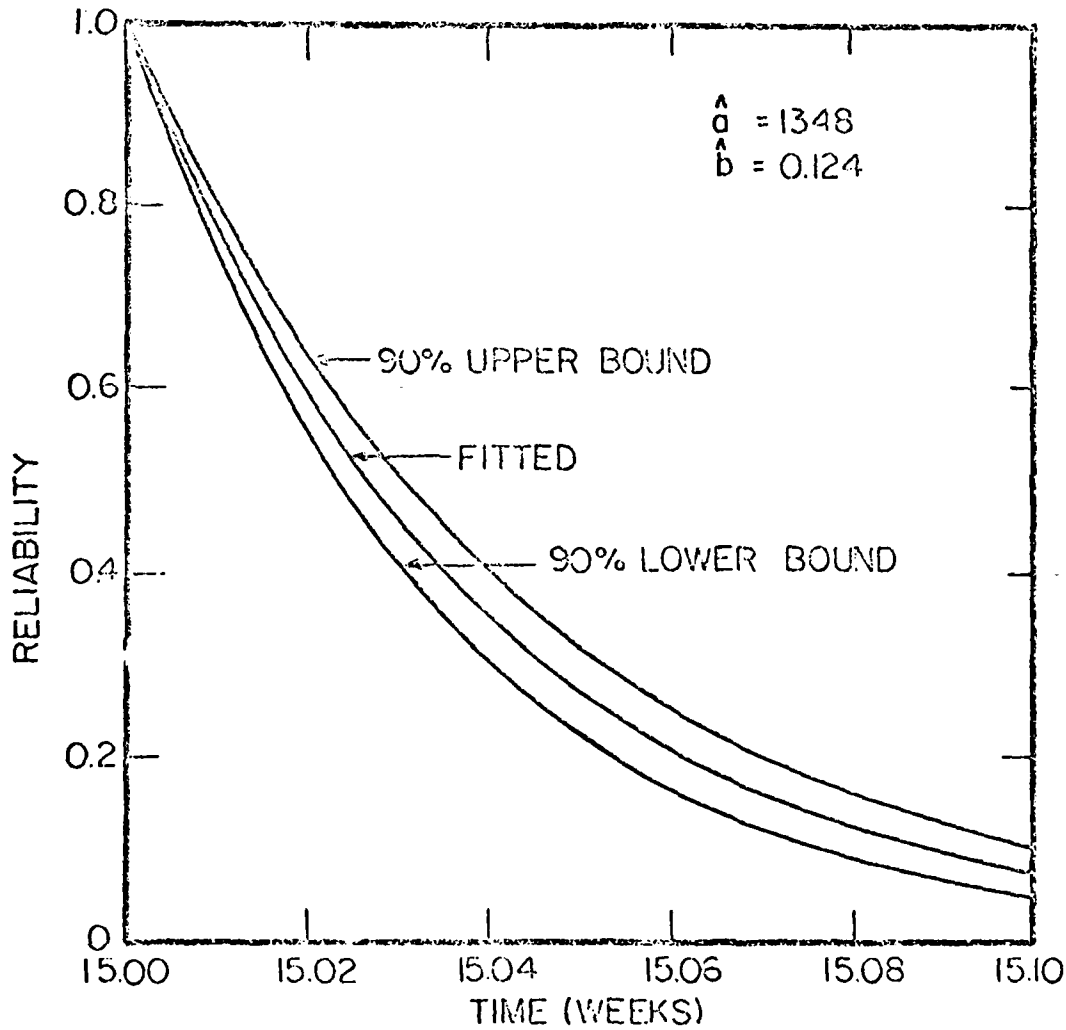


Figure 7.5 Reliability and 90% confidence bounds after 15 weeks of testing

7.8 Summary of Analyses for DS1 to DS4

Analyses similar to those for data set DS1 were undertaken for data sets DS2, DS3 and DS4 of Table 7.3. A summary of the results is given in Table 7.5.

TABLE 7.5

A SUMMARY OF DATA ANALYSES

Quantity \ Data Set	DS1	DS2	DS3	DS4
$\hat{a}$	1348	1823	3958	3446
$\hat{b}$	0.124	0.112	0.0768	0.0771
$\sqrt{\widehat{Var}(\hat{a})}$	48.7	62.2	147.3	136.6
$\sqrt{\widehat{Var}(\hat{b})}$	0.00745	0.00643	0.00460	0.00492
$\hat{\rho}_{a,b}$	-0.571	-0.648	-0.856	-0.855
Estimated Number of Remaining Errors at the end of Operational Demonstration	209	338	1212	1050
Number of Errors Detected During Nine Months of Operation	198	263	540	475

8. ANALYSIS OF FAILURE DATA FROM NAVAL TACTICAL DATA SYSTEM (NTDS)

Jelinski and Moranda (Reference 31) first analyzed some software failure data from the U.S. Navy Fleet Computer Programming Center. Since then this data set has been used by several investigators for model validation purposes. In this section we analyze the same data set to see how good the NHPP model is in modelling these failures. In the next section we will compare the results from the NHPP model with those of Jelinski and Moranda.

The data set was extracted from information about errors in the development of software for the real-time, multi-computer complex which forms the core of the Naval Tactical Data System (NTDS). The NTDS software consisted of some 38 different project schedules. Each module was supposed to follow three stages: the production (or development) phase, the test phase, and the user phase. Many of the "trouble reports" or "software anomaly reports" were generated whenever a system-level symptom of a deficiency was noted by operators or users. A proper trace back to the exact cause in software of this symptom was done by personnel familiar with the entire system. However, Jelinski and Moranda felt that it was better to analyze the data from isolated modules than from the total system, due to the fact that many of the modules did not evolve in the fashion indicated. One of the larger modules, denoted by A-module, had the desired pattern. The times (in days) between failures for this module are shown in Table 8.1. Twenty-six software errors were found during the production phase and five

TABLE 8.1

SOFTWARE FAILURE DATA FROM NTDS

ERROR NO.	TIME BETWEEN FAILURES x_k, days	CUMULATIVE TIME $s_n = \Sigma x_k$, days
Production (Checkout) Phase		
1	9	9
2	12	21
3	11	32
4	4	36
5	7	43
6	2	45
7	5	50
8	8	58
9	5	63
10	7	70
11	1	71
12	6	77
13	1	78
14	9	87
15	4	91
16	1	92
17	3	95
18	3	98
19	6	104
20	1	105
21	11	116
22	33	149
23	7	156
24	91	247
25	2	249
26	1	250
Test Phase		
27	87	337
28	47	384
29	12	396
30	9	405
31	135	540
User Phase		
32	258	798
Test Phase		
33	16	814
34	35	849

additional errors during the test phase. The last error was found on 4 Jan 1971. One error was observed during the user phase on 20 Sept 1971 and two more errors (5 Oct 1971, 10 Nov 1971) during the test phase. This indicates that a re-work of the module had taken place after the user error was found. A more detailed description of the NTDS software can be found in Jelinski and Moranda.

Data Analyses

The data in this case is available as times between software failures and hence the method described in Section 4.2 will be used for estimation of parameters. We consider the first 26 data points in Table 8.1, for which n=26 and $s_{26} = \sum_{k=1}^{26} x_k = 250$ days.

To get an appreciation of the likelihood function associated with this data set, the log-likelihood from Equation (4.20) is plotted in Figure 8.1. We see that the surface rises sharply along the b-axis and is relatively flat along the a-axis.

The maximum of this surface is obtained by solving Equations (4.23) and (4.24). Substituting the appropriate values from Table 8.1 in Equations (4.23) and (4.24) we get

$$\frac{26}{a} = 1 - e^{-b(250)} \tag{8.1}$$

and

$$\frac{26}{b} = a(250)\cdot e^{-b(250)} - b(250) . \tag{8.2}$$

Solving Equations (8.1) and (8.2), numerically, we get

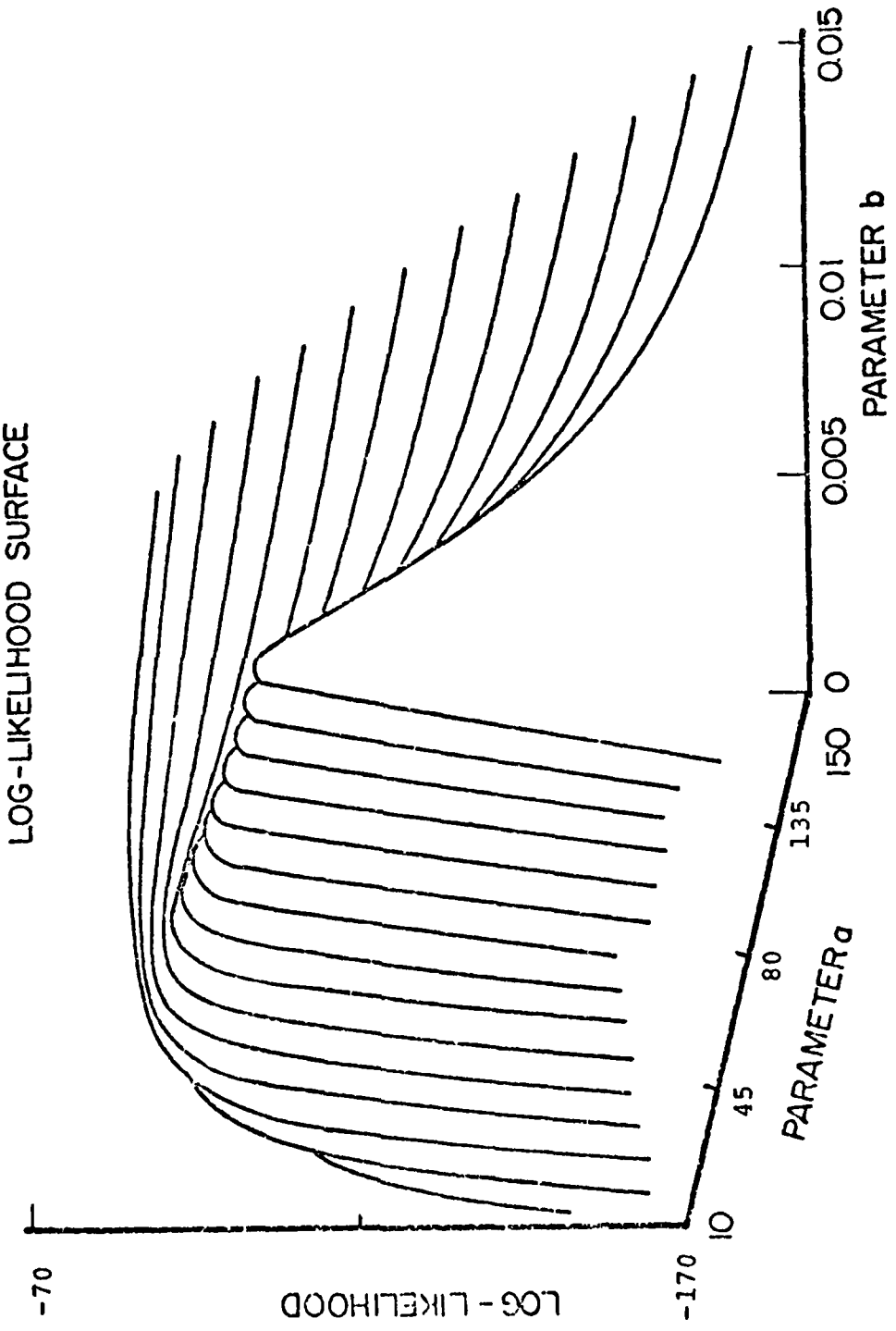


Figure 8.1 Log-Likelihood Surface Based on NTDS Data.

$$\hat{a} = 33.99$$

and

$$\hat{b} = 0.00579$$

as the mle's for a and b, respectively. The fitted mean value function is

$$\hat{m}(t) = 33.99(1 - e^{-0.00579t}) . \qquad (8.3)$$

and is shown in Figure 8.2, along with the actual data (determination of the confidence bounds will be discussed later).

Goodness-of-fit test

We now perform the Kolmogorov-Smirnov goodness-of-fit test to check the adequacy of the fitted model. Now, using Corollary 5.1 and the results in Section 5, we conduct the test based on $26-1 = 25$ points. The hypothesis, from Equation (5.2), is

$$H_0 : G_0(x) = \frac{1-e^{-b_0 x}}{1-e^{-b_0(250)}} \quad \text{for} \quad 0 \le x \le 250 , \qquad (8.4)$$

and the sample cdf is

$$H(x) = \begin{cases} 0 & , \ x < s_1 \\ k/25 & , \ s_{k-1} \le x < s_k , \\ 1 & , \ x \ge s_{25} \end{cases} \quad k = 2,3,\ldots,25 . \qquad (8.5)$$

The values of s_k and $H(s_k)$ are given in Table 8.2. To compute $G_0(s_k)$ for various s_k values, we replace b_0 by $\hat{b}$ in Equation (8.4)

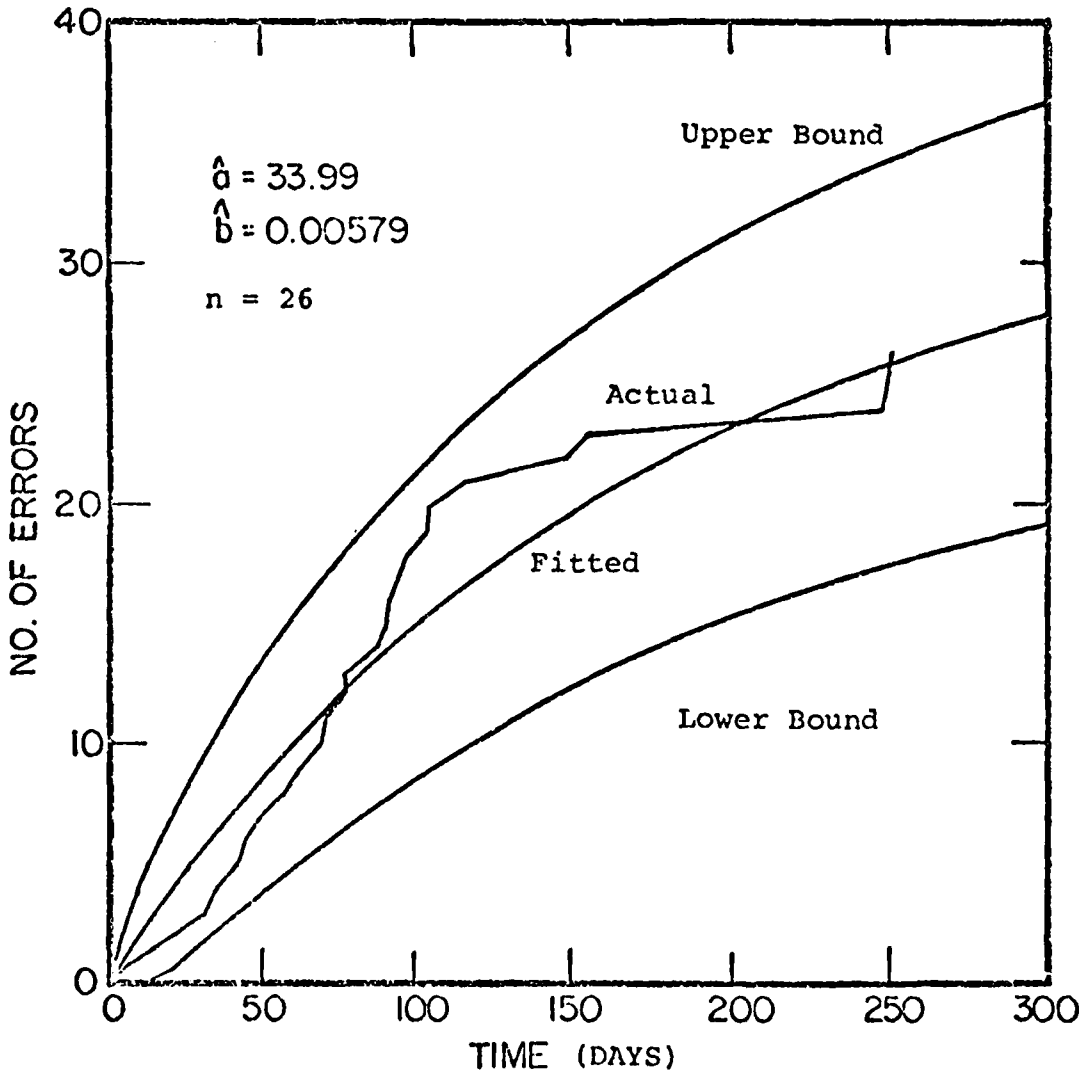


Figure 8.4 Plots of Mean Value Function and 90% Confidence Bounds for the N(t) Process (NTDS Data)

TABLE 8.2

KOLMOGOROV-SMIRNOV TEST
FOR THE NTDS DATA SET

k	s_k	$H(s_k)$	$G_0(s_k)$	$\lvert G_0(s_k)-H(s_k)\rvert$	$\lvert G_0(s_k)-H(s_{k-1})\rvert$
1	9	0.04	0.0664	0.0264	0.0664
2	21	0.08	0.1497	0.0697	0.1097
3	32	0.12	0.2211	0.1011	0.1411
4	36	0.16	0.2460	0.0860	0.1260
5	43	0.20	0.2882	0.0882	0.1282
6	45	0.24	0.2999	0.0599	0.0999
7	50	0.28	0.3286	0.0486	0.0886
8	58	0.32	0.3730	0.0530	0.0930
9	63	0.36	0.3996	0.0396	0.0796
10	70	0.40	0.4357	0.0357	0.0757
11	71	0.44	0.4407	0.0007	0.0407
12	77	0.48	0.4703	0.0097	0.0303
13	78	0.52	0.4751	0.0449	0.0049
14	87	0.56	0.5174	0.0426	0.0026
15	91	0.60	0.5355	0.0645	0.0245
16	92	0.64	0.5399	0.1001	0.0601
17	95	0.68	0.5532	0.1268	0.0868
18	98	0.72	0.5661	0.1539	0.1139
19	104	0.76	0.5915	0.1685	0.1285
20	105	0.80	0.5956	0.2044	0.1644
21	116	0.84	0.6395	0.2005	0.1605
22	149	0.88	0.7557	0.1243	0.0843
23	156	0.92	0.7776	0.1424	0.1024
24	247	0.96	0.9946	0.0346	0.0746
25	249	1.00	0.9982	0.0018	0.0382

and obtain Column 4 of Table 8.2. Entries in Columns 5 and 6 are easily obtained from Columns 3 and 4. Now, from Equations (5.5) and (8.4)

$$D = \max_k \{ |G_0(s_k) - H(s_k)|, |G_0(s_k) - H(s_{k-1})| \} .$$

In other words, D is the largest entry in Columns 5 or 6 and is seen to be

$$D = 0.2044 .$$

To test at $\alpha = .05$, we use a critical value corresponding to $\alpha = .20$ as discussed in Section 5.

From statistical tables,

$$D_{25;0.2} = 0.208 .$$

Since $D < D_{25;0.2}$, we accept the null hypothesis, H_0, at 5% level of significance.

The $100(1-\alpha)\%$ confidence limits for $G(x)$ can now be calculated from Equation (5.6). For example, for $\alpha = 0.05$ we have $D_{25;0.05} = 0.264$, so that the lower and upper confidence bounds are

$$L(x) = \max\{H(x) - 0.264, 0\}$$

and

$$U(x) = \min\{H(x) + 0.264, 1\} ,$$

where $H(x)$ is given by Equation (8.5). The 95% bounds for $G(x)$, along with $G_0(x)$, are shown in Figure 8.3. We see that the fitted model seems to be adequate.

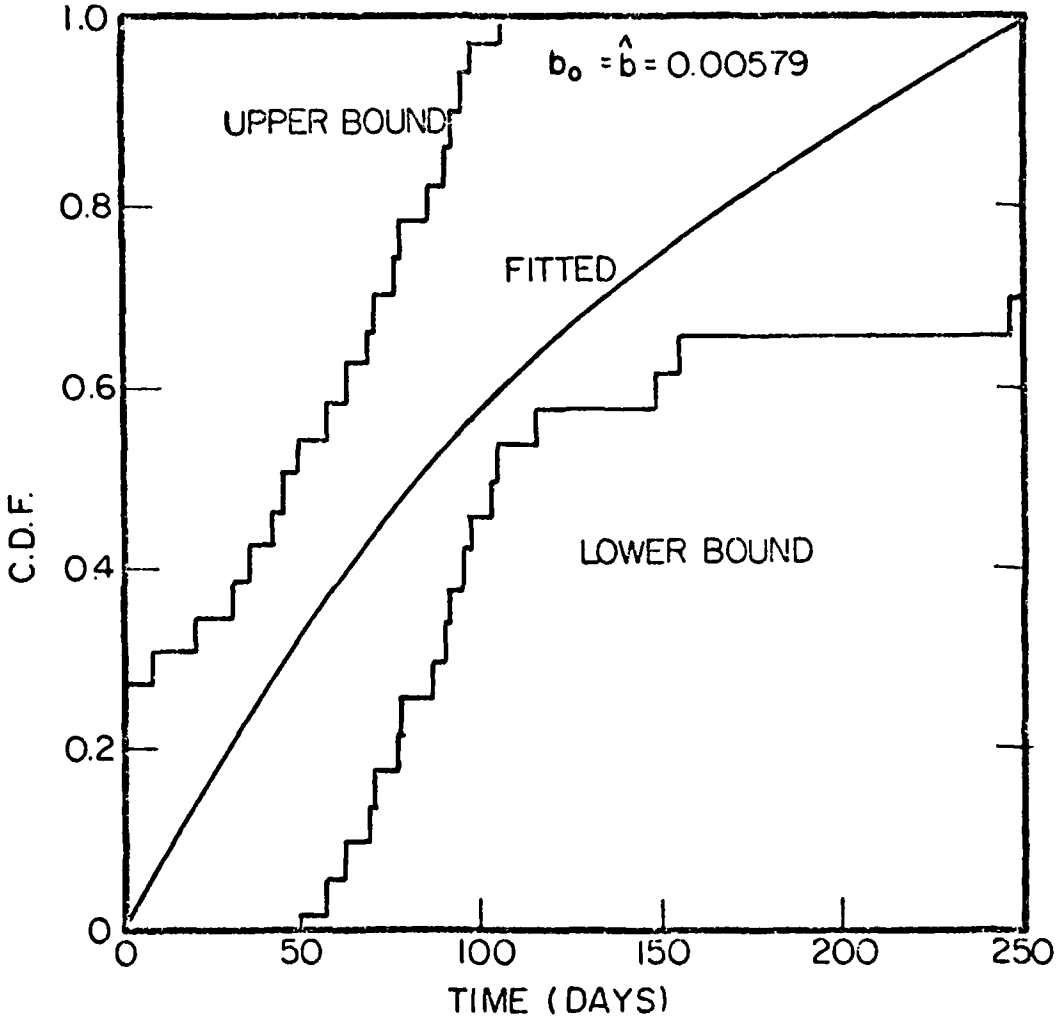


Figure 8.3 95% confidence bounds for the conditional c.d.f. G(x) and the fitted C.D.F. curve (NTDS data)

Having established that the model provides a good fit, various performance measures of interest can be obtained by substituting the estimated values of a and b in the appropriate equations of Section 3.

The estimated mean value function, as given in Equation 8.3, is $\hat{m}(t) = 33.99(1-3^{-0.00579t})$. A plot of $\hat{m}(t)$ and the actual number of errors detected during the production period for this case was given in Figure 8.2. Also shown were the 90% confidence bounds for the $N(t)$ process as computed from Equation (3.1).

The $100(1-\alpha)\%$ confidence regions for a and b are obtained from Equations (4.26) and (4.27) following a procedure similar to the one detailed in Section 7. These are shown in Figure 8.4 for $\alpha = 0.05$, 0.25, and 0.50.

Next, an estimate of the expected number of errors remaining in the software system at $t = 250$ days, given that $N(250) = 26$, is obtained from Equation (3.8) as

$$E[\bar{N}(250) | N(250)=26] = 33.99 - 26 = 7.99 .$$

As indicated in Table 8.1, eight errors were found during usage of the system, subsequent to the production phase. The excellent match between the predicted and actual values is coincidental and in general the NHPP model is not expected to perform this well.

Finally, software reliability, $R_{X_{27}|s_{26}}(x|250)$, can be computed from Equation (3.19). For example, the reliability values after $x = 5$, 10, 20 and 30 days are 0.796, 0.638, 0.417 and 0.280, respectively. Thus the probability that the system will operate

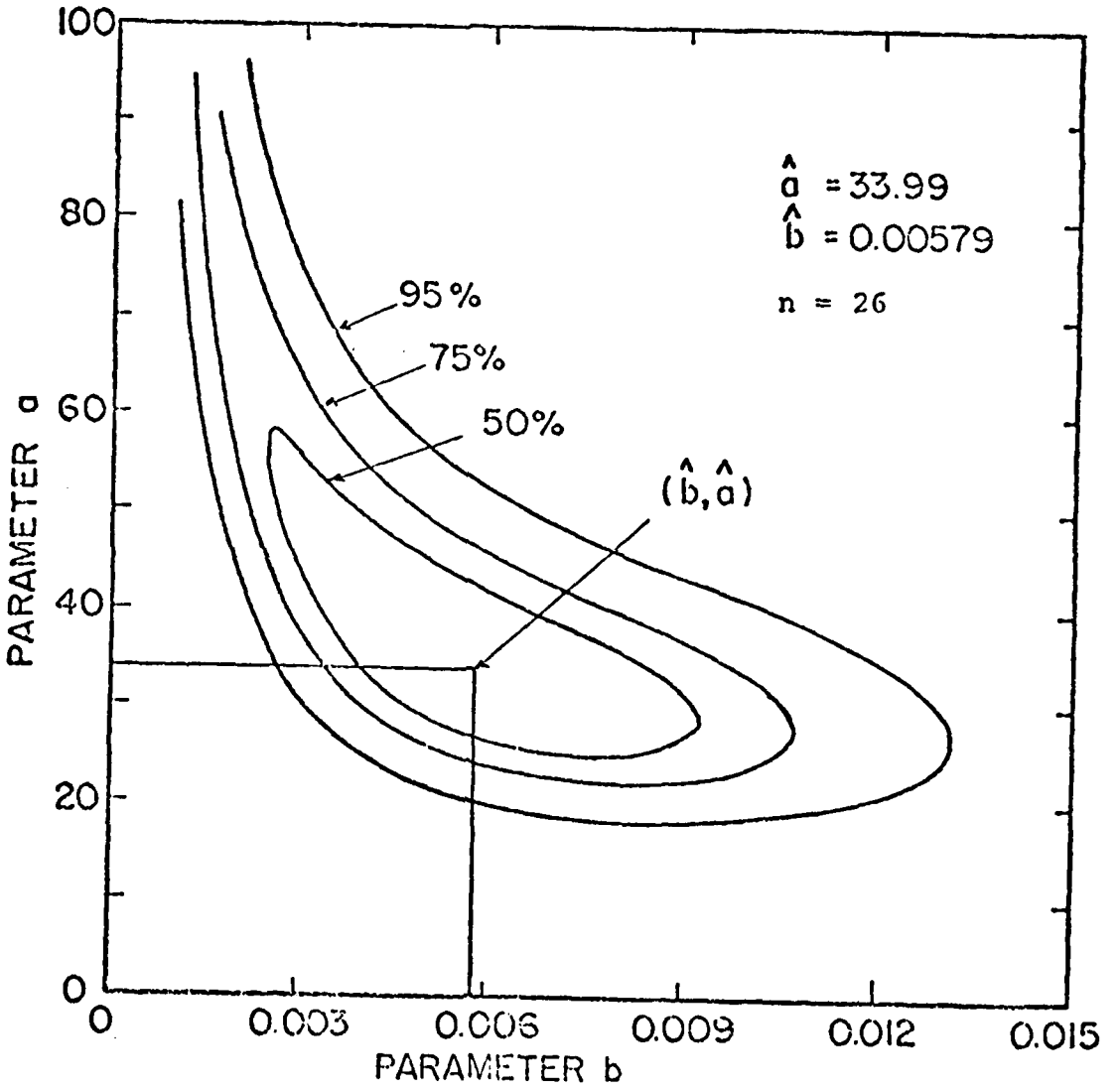


Figure 8.4 Joint Confidence Regions for a and b (NTDS Data)

without any failures for 30 additional days in 0.26. As seen from the data in Table 8.1, the system did operate without any failures for 87 days subsequent to failure number 26.

9. A COMPARISON OF NTDS DATA ANALYSES USING THE NHPP AND THE DE-EUTROPHICATION MODELS

As mentioned in Section 8, the NTDS data has been analyzed previously by several investigators for model validation purposes. The first such analysis was undertaken by Jelinski and Moranda (Reference 31) using a De-Eutrophication model. In this section we provide a limited comparison of the results of analyses using the NHPP and the De-Eutrophication models.

For the De-Eutrophication process, the cdf of X_k, the time between the (k-1)st and the kth failures, is given by

$$\begin{aligned} \Phi_k(x) &= F_{N-k+1}(x) \\ &= 1 - e^{-(N-k+1)\varphi x}, \quad k = 1,2,\ldots, \end{aligned} \tag{9.1}$$

where N is the number of errors in the system at time zero and φ is the error occurrence rate per error.

The likelihood function for N and φ for given data $x_1, x_2, \ldots, x_n$ is

$$\begin{aligned} L(N,\varphi|\underline{x}) &= \prod_{k=1}^{n} f_{N-k+1}(x_k) \\ &= \prod_{k=1}^{n} (N-k+1)\varphi e^{-(N-k+1)\varphi x_k} \end{aligned}$$

and the log-likelihood is

$$\ell n L(N,\varphi|\underline{x}) = n\ell n\varphi + \sum_{k=1}^{n} \ell n(N-k+1) - \sum_{k=1}^{n} (N-k+1)\varphi x_k . \tag{9.2}$$

The maximum likelihood estimates of N and φ are the values $\hat{N}$ and $\hat{\varphi}$, respectively, that maximize Equation (9.2). Taking the partial derivatives of Equation (9.2) and equating them to zero, the likelihood equations which the mle's must satisfy are

$$\varphi \sum_{k=1}^{n} x_k = \sum_{k=1}^{n} \frac{1}{N-k+1} \tag{9.3}$$

$$n/\varphi = \sum_{k=1}^{n} (N-k+1)x_k \, . \tag{9.4}$$

For the first 26 points in Table 8.1, the solutions of Equations (9.3) and (9.4) are $\hat{N} = 31.2$ and $\hat{\varphi} = 0.00685$. In other words, the initial number of errors is estimated to be 31.2 and the failure rate is estimated to be 0.00685 errors per error day. Therefore, the estimated number of errors remaining at the end of the production phase (i.e., at $t = 250$) is $\hat{N} - 26 = 5.2$.

The estimates of a and b at $t = 250$, as obtained in Section 8, were $\hat{a} = 33.99$ and $\hat{b} = 0.00579$.

It can be easily shown that for the De-Eutrophication process the expected number of errors detected by time t is given by

$$M_N(t) = N(1 - e^{-\varphi t}) \, . \tag{9.5}$$

Substituting for N and φ,

$$\hat{M}_N(t) = 31.2(1 - e^{-0.00685t}) \, . \tag{9.6}$$

For the $N(t)$ process (NHPP model), the expected number of errors by time t, as given in Equation (8.3), is

$$E[N(t)] = \hat{m}(t) = 33.99(1 - e^{-0.00579t}) \, . \tag{9.7}$$

Equations (9.6) and (9.7) represent the same physical quantity. Plots of $\hat{M}_N(t)$ and $\hat{m}(t)$ are shown in Figure 9.1. The actual number of errors detected by time t is also shown. A comparison of the plots shows that results for the NIIPP and De-Eutrophication processes are quite close.

The mean time to the kth failure (after the $(k-1)$st failure) is the reciprocal of the parameter $(N-k+1)\varphi$ in Equation (9.1), i.e.,

$$E[X_k] = \frac{1}{(N-k+1)\varphi}, \qquad k = 1,2,\ldots, \tag{9.8}$$

or

$$E[X_k] = \frac{1}{(31.2-k+1)0.00685}. \tag{9.9}$$

The values for $k = 1,2,\ldots,31$ were computed and are shown in Table 9.1. As was pointed out in Section 4.2, the MTTF for the $N(t)$ process does not exist due to the fact that the distribution of X_k is improper. For the sake of comparison, however, we use the inverse transformation of the mean value function to get the estimate of time to kth failure as follows.

$$\begin{aligned}\hat{s}_k &= \hat{m}^{-1}(k) \\ &= -\frac{1}{\hat{b}}\ell n(1-k/\hat{a}) \\ &= -\frac{1}{0.00579}\ell n(1-k/33.99), \qquad k = 1,2,\ldots.\end{aligned}$$

Hence, we get

$$\hat{x}_k = \hat{s}_k - \hat{s}_{k-1}. \tag{9.10}$$

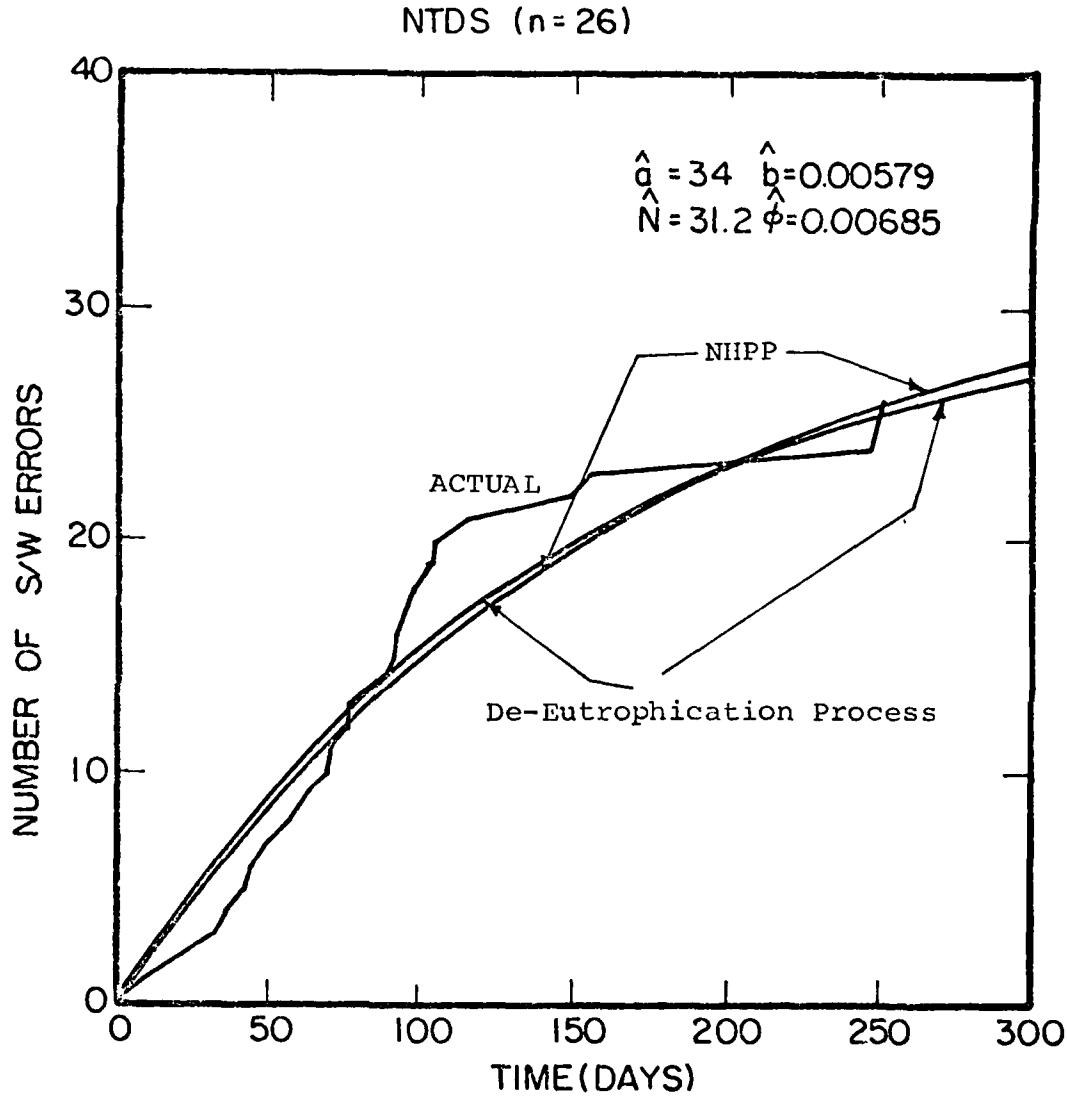


FIGURE 9.1 Plots of the NTDS data and the estimated mean value functions $\hat{M}_N(t)$ and $\hat{m}(t)$.

TABLE 9.1

Comparison of Results Based on the NHPP and the De-Eutrophication Models

ERROR NO. (k)	ACTUAL FAILURE TIME (DAYS) (x_k)	ESTIMATED FAILURE TIME (DAYS), $\hat{x}_k$ (Based on 26 observations)	
		NHPP	Jelinski-Moranda Model
1	9	5.156	4.679
2	12	5.315	4.834
3	11	5.483	5
4	4	5.663	5.177
5	7	5.855	5.367
6	2	6.061	5.572
7	5	6.281	5.793
8	8	6.518	6.032
9	5	6.774	6.292
10	7	7.05	6.576
11	1	7.351	6.886
12	6	7.677	7.227
13	1	8.035	7.603
14	9	8.427	8.021
15	4	8.859	8.488
16	1	9.338	9.011
17	3	9.872	9.604
18	3	10.471	10.281
19	6	11.147	11.06
20	1	11.916	11.966
21	11	12.799	13.034
22	33	13.824	14.312
23	7	15.028	15.868
24	91	16.461	17.803
25	2	18.197	20.276
26	1	20.342	23.546
27	87	23.062	28.074
28	47	26.624	34.758
29	12	31.489	45.62
30	9	38.539	66.357
31	135	49.686	121.655
Total No. of Errors (34)		$\hat{a} = 34$	$\hat{N} = 31.2$
Failure Rate		$\hat{b} = 0.00579$	$\hat{\varphi} = 0.00685$
No. of Remaining Errors (8)		$\hat{a} - 26 = 8$	$\hat{N} - 26 = 5.2$

The values of $\hat{x}_k$ for $k=1,2,\ldots,31$ were computed from Equation (9.10) and are given in Table 9.1. As a criterion for comparing the results from the two models, we choose the sums of squares of the differences between the actual values x_k and the estimated values $\hat{x}_k$ for $k=1,2,\ldots,26$ and for $k=27,28,\ldots,31$, i.e., we use

$$\text{Fit} = \sum_{k=1}^{26} (x_k - \hat{x}_k)^2 ,$$

and

$$\text{Prediction} = \sum_{k=27}^{31} (x_k - \hat{x}_k)^2 .$$

We get Fit $=7169$ and Prediction $=8220$ for the De-Eutrophication process. For the N(t) process we get Fit $=7180$ and Prediction $=13034$. For this criterion, the De-Eutrophication process gives better results than the N(t) process. However, NHPP gives better results when the criterion is the number of errors remaining at $t=250$ days. These results are summarized in Table 9.1.

Next we compare the accuracy of estimates $(\hat{N},\hat{\varphi})$ with that of $(\hat{a},\hat{b})$ by obtaining joint confidence regions for (N,φ) and (a,b).

The joint $100(1-\alpha)\%$ confidence regions for (a,b) are given by Equations (4.26) and (4.27) and were shown in Figure 8.4 for $\alpha = .05, .25$ and $.50$.

To obtain the $100(1-\alpha)\%$ joint confidence regions for N and φ, we use the same result that was used to get the confidence regions for (a,b), viz.

$$\ell nL(\hat{N},\hat{\varphi}|\underline{x}) - \ell nL(N,\varphi|\underline{x}) = \frac{1}{2}\chi^2_{2;\alpha} . \tag{9.11}$$

Substituting in Equation (9.11) the expression for $\ell nL(N,\varphi|\underline{x})$ from Equation (9.2), we get

$$n\ell n\varphi + \sum_{k=1}^{n} \ell n(n-k+1) - \sum_{k=1}^{n} (N-k+1)\varphi x_k = c , \tag{9.12}$$

where

$$c = \ell nL(\hat{N},\hat{\varphi}|\underline{x}) - \frac{1}{2}\chi^2_{2;\alpha} . \tag{9.13}$$

Equation (9.12) defines a joint $100(1-\alpha)\%$ confidence region for N and φ. Plots of such regions for $\alpha = .05$, $.25$ and $.50$ are shown in Figure 9.2. A comparison of these plots with those in Figure 8.4 shows that for the same α, the range of N is somewhat smaller than that of a while the range of φ is larger than that of b.

Next we compare the reliability predictions based on the two models. The reliability function after n errors is

$$R_{n+1}(x) = 1-P[(n+1)\text{st error will occur by time } x | n \text{ errors have occurred}] . \tag{9.14}$$

For the De-Eutrophication process of Equation (9.1), Equation (9.14) becomes

$$R_{n+1}(x) = e^{-(N-n)\varphi x} . \tag{9.15}$$

Since after $n = 26$, $\hat{N} = 31.2$, $\hat{\varphi} = 0.00685$, we have

$$\hat{R}_{27}(x) = e^{-(31.2-26)(0.00685)x}$$

or

$$\hat{R}_{27}(x) = e^{-0.0356x} . \tag{9.16}$$

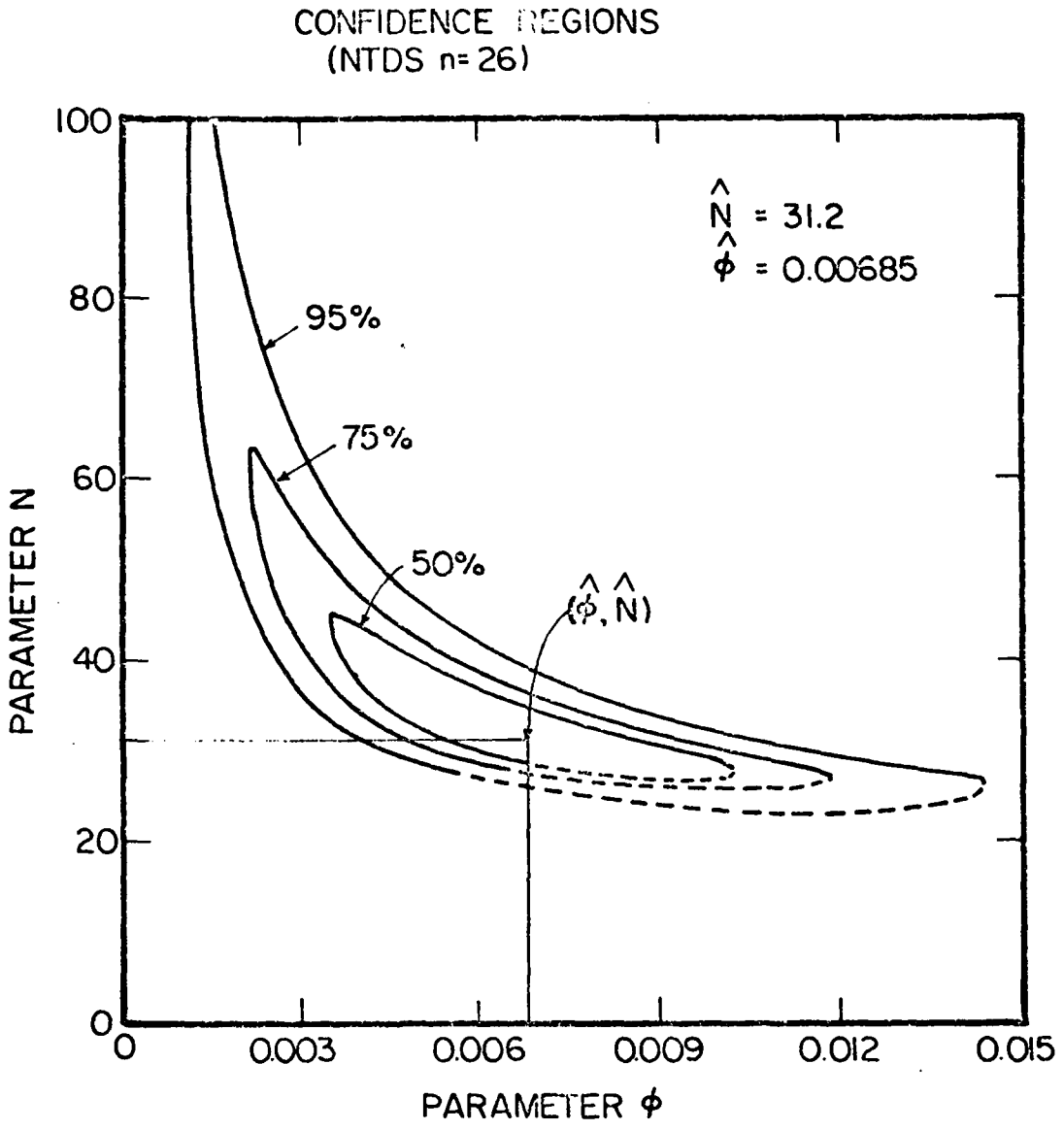


Figure 9.2 Joint confidence regions for N and φ (NTDS Data)

For the NHPP, the reliability function from Equation (3.19) is

$$\hat{R}_{X_{n+1}|S_n}(x|s) = e^{-\hat{a}\{e^{-\hat{b}s} - e^{-\hat{b}(s+x)}\}} .$$

Since after $n = 26$, $\hat{a} = 33.99$, $\hat{b} = 0.00579$, we get

$$\hat{R}_{X_{27}|S_{26}}(x|250) = e^{-33.99\{e^{-(0.00579)(250)} - e^{-(0.00579)(250+x)}\}}$$

or

$$\hat{R}_{X_{27}|S_{26}}(x|250) = e^{-33.99 \cdot e^{-(.00579)250}(1-e^{-.00579x})}$$

or

$$\hat{R}_{X_{27}|S_{26}}(x|250) = e^{-7.993(1-e^{-0.00579x})} . \qquad (9.17)$$

Plots of $\hat{R}_{27}(x)$ and $\hat{R}_{X_{27}|S_{26}}(x|250)$ for various values of x are shown in Figure 9.3. Also shown are the reliability functions $\hat{R}_{32}(x)$ and $\hat{R}_{X_{32}|S_{31}}(x|540)$ computed from the data for the first 31 failures given in Table 9.1. The reliability after $n = 31$ is monotonically higher than that after $n = 26$. Also, the predictions from NHPP are somewhat more conservative than those from the De-Eutrophication process. This is what would be expected because of the larger and more accurate estimates of the number of errors remaining in the system when the NHPP model is employed.

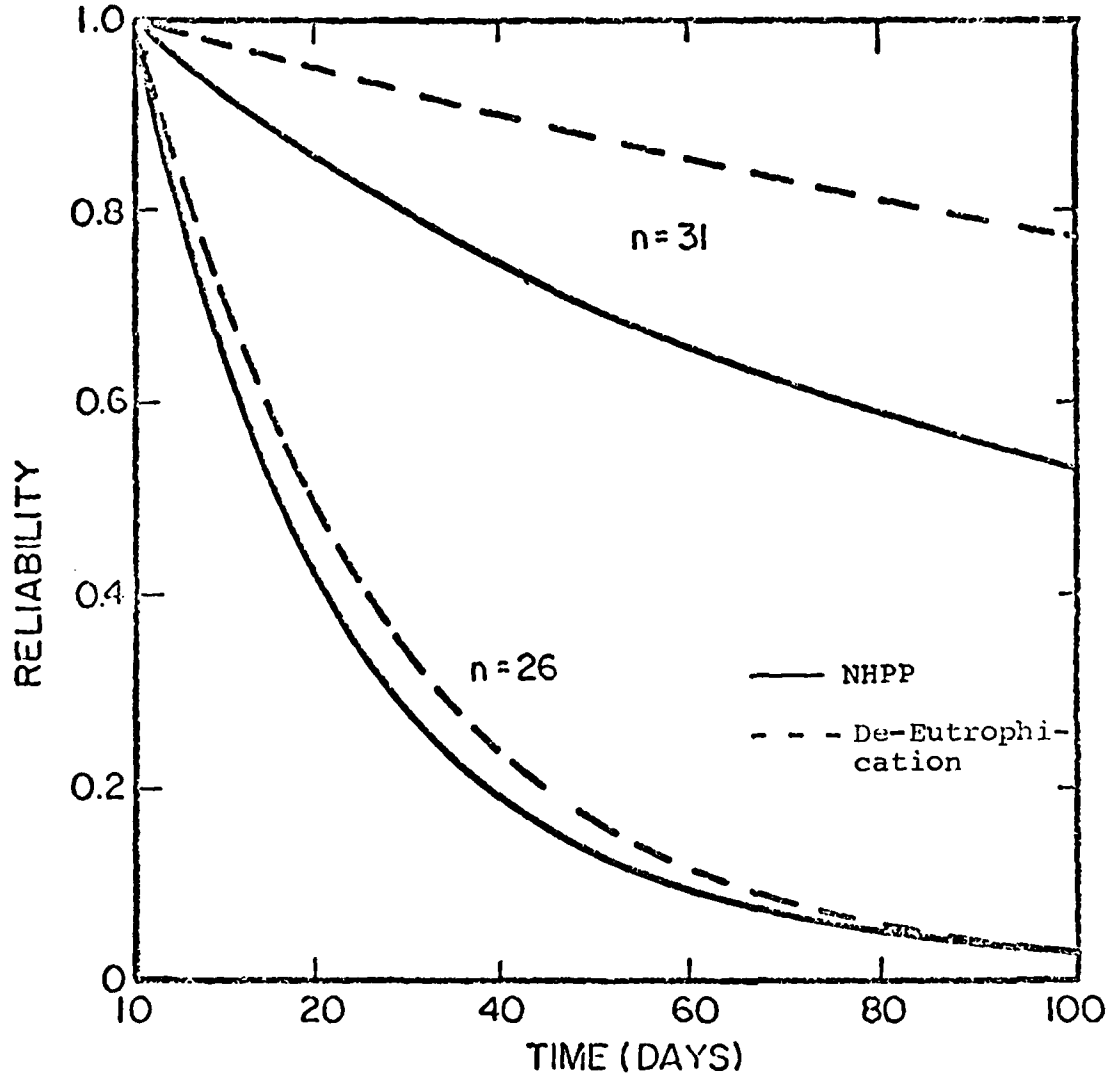


Figure 9.3 Plots of Reliability Functions Based on NHPP and De-Eutrophication Models.

10. CONCLUSIONS

In this report we proposed a simple but very versatile model for analyzing failures in large scale software systems. The model was justified on the basis of reasonable and realistic assumptions about the nature of the failure phenomenon. Specifically, the model (Section 2) is based on a non-homogeneous Poisson process (NHPP) with a mean value function $m(t) = a(1 - e^{-bt})$. The choice of the form of this mean value function was also justified.

The parameters of this model are a and b, where a is the expected number of failures that will be encountered if the system were to be used for a long time and b is the error detection rate per error.

Several useful quantitative measures were proposed (Section 3) for assessing software performance. These measures are the number of failures by time t, number of errors remaining in the system at t, software reliability, etc. Models were also developed for computing these measures from actual failure data.

A methodology for obtaining the maximum likelihood estimates of a and b was presented (Section 4) for the cases when the data is given as failure counts or as times between failures. A goodness-of-fit test based on the Kolmogorov-Smirnov statistic was developed (Section 5) to check the adequacy of the fitted model.

Failure data from two DOD systems were analyzed (Sections 7 and 8) using the methodology presented here (Section 6). Results of the analyses and a limited comparative study (Section 9) indicate that the NHPP model seems to do quite well in explaining the failure occurrence phenomenon. Applications of this model to several other data sets, not reported here, also yielded satisfactory results.

The model developed in this report is applicable after the system error occurrence rate begins to decline. At present, all available models share this restriction. Efforts are under way to develop a 3-parameter NHPP model which will be applicable during the integration phase. Also, the parameters cannot be estimated without available data. Work is continuing on the development of a Bayesian methodology which will permit determination of a and b when limited data is available.

REFERENCES

1. Abramowitz, M. and Stegun, I.A., Handbook of Mathematical Functions, Dover Publications, Inc., 1965.

2. Allen, A.O., Probability, Statistics and Queueing Theory, Academic Press, 1978.

3. Angus, J.E., Schafer, R.E., and Sukert, A., "Software Reliability Model Validation," Proceedings of Annual Reliability and Maintainability Symposium, San Francisco, California, January 1980, pp. 191-193.

4. Barlow, R.E. and Proschan, F., Statistical Theory of Reliability and Life Testing: Probability Models, Holt, Rinehart and Winston, Inc., 1975.

5. Basin, S.L., Estimation of Software Error Rate Via Capture-Recapture Sampling, Science Applications, Inc., Palo Alto, California, 1974.

6. Boehm, B.W., "Software Engineering," IEEE Trans. on Computers, Vol. C-25, No. 12, December 1976, pp. 1226-1241.

7. Brooks, W.D. and Motley, R.W., Analysis of Discrete Software Reliability Models, IBM, Final Technical Report, RADC-TR-80-84, (in print).

8. Brown, J.R. and Lipow, M., "Testing for Software Reliability," Proceeding, International Conference on Reliable Software, Los Angeles, April 1975, pp. 518-527.

9. Brown, M., "Statistical Analysis of Non-Homogeneous Poisson Processes," in Stochastic Point Processes, edited by P.A.W. Lewis, Wiley, 1972, pp. 67-89.

10. Buck, R.C., Advanced Calculus, McGraw-Hill Book, Co., 1956.

11. Cox, D.R. and Lewis, P.A.W., The Statistical Analysis of Series of Events, Methuen, London, 1966.

12. Cox, D.R. and Miller, H.D., The Theory of Stochastic Processes, Wiley and Sons, Inc., 1965.

13. Crow, L.H., "Reliability Analysis for Complex, Repairable Systems," Reliability and Biometry, edited by F. Proschan and R.J. Serfling, SIAM, 1974, pp. 379-410.

14. Dickson, J.D., Hesse, J.L., Kientz, A.C. and Shooman, M.L., "Quantitative Analysis of Software Reliability," Proceedings, Annual Reliability and Maintainability Symposium, New York, January 1972, pp. 148-157.

15. Donelson, J., III, Duane's Reliability Growth Model as a Non-Homogeneous Poisson Process, IDA Log. No. HQ76-18012, Paper P-1162, December, 1975.

16. Duane, J.T., "Learning Curve Approach to Reliability Monitoring," IEEE Trans. Aerospace, Vol. 2, April 1964, pp. 563-566.

17. Endres, A., "An Analysis of Errors and Their Causes in System Programs," Proceedings International Conference on Reliable Software, Los Angeles, California, April 1975, pp. 327-336.

18. Feller, W., An Introduction to Probability Theory and Its Applications, 2nd Ed., Vol. I, Wiley, 1957.

19. Feller, W., An Introduction to Probability Theory and Its Applications, Vol. II, Wiley, 1966.

20. Finkelstein, J.M., "Confidence Bounds on the Parameters of the Weibull Process," Technometrics, Vol. 18, No. 1, March 1976, pp. 115-117.

21. Forman, E.H. and Singpurwalla, N.D., "An Empirical Stopping Rule for Debugging and Testing Computer Software," Journal of the American Statistical Association, Vol. 72, No. 360, December 1977, pp. 750-757.

22. Fries, M.J., Software Error Data Acquisition, Boeing Aerospace Co., Final Technical Report, RADC-TR-77-130, April 1977, AD A039-916.

23. Goel, A.L., Summary of Technical Progress: Bayesian Software Reliability Prediction Models, RADC-TR-77-112, Syracuse University, March 1977, AD A039-022.

24. Goel, A.L., "Reliability and Other Performance Measures of Computer Software," Proceedings, First International Conference on Reliability and Exploitation of Computer Systems, Wroclaw, Poland, September 1979, pp. 23-31.

25. Goel, A.L., "A Software Error Detection Model with Applications," Journal of Systems and Software, (to appear), 1980.

26. Goel, A.L. and Okumoto, K., An Imperfect Debugging Model for Software Reliability, Final Technical Report, Syracuse University, RADC-TR-78-155, Vol. 1 (of 5), July 1978, AD A057-879.

27. Goel, A.L. and Okumoto, K., Bayesian Software Correction Limit Policies, Final Technical Report, Syracuse University, RADC-TR-78-155, Vol. 2 (of 5), July 1978, AD A057-872.

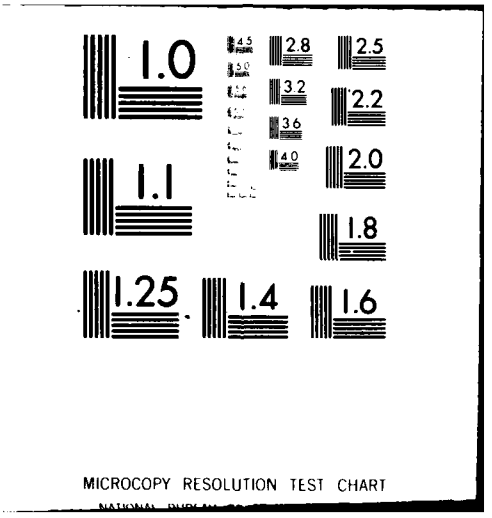
28. Goel, A.L. and Okumoto, K., "An Analysis of Recurrent Software Failures in a Real-Time Control System," Proceedings Annual Technical Conference, ACM, Washington, D.C., December 1978, pp. 496-500.

AD-A088 186 SYRACUSE UNIV N Y DEPT OF INDUSTRIAL ENGINEERING AND--ETC F/G 12/1
A TIME DEPENDENT ERROR DETECTION RATE MODEL FOR SOFTWARE PERFOR--ETC(U)
MAY 80 A L GOEL, K OKUMOTO F30602-78-C-0351

UNCLASSIFIED RADC-TR-80-179 NL

2 OF 2
AD A088186

END
DATE FILMED
9-80
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
MICROCOPY RESOLUTION TEST CHART

29. Goel, A.L. and Okumoto, K., "A Time Dependent Error Detection Rate Model for a Large Scale Software System," Proceedings Third USA-Japan Computer Conference, San Francisco, California, October, 1978, pp. 35-40.

30. Goel, A.L. and Okumoto, K.,"A Markovian Model for Reliability and Other Performance Measures of Software Systems," Proceedings National Computer Conference, New York, Vol. 48, June 1979, pp. 769-774.

31. Jelinski, Z. and Moranda, P., "Software Reliability Research," in Statistical Computer Performance Evaluation, W. Freiberger (ed.), Academic Press, 1972, pp. 465-484.

32. Lewis, P.A.W., Implications of a Failure Model for the Use and Maintenance of Computers," Journal of Applied Probability, Vol. 1, 1964, pp. 347-368.

33. Lewis, P.A.W. and Shedler, G.S., "Statistical Analysis of Non-stationary Series of Events in a Data Base System," IBM Journal of Research and Development, Vol. 20, September 1976, pp. 465-482.

34. Lipow, M., Estimation of Software Package Residual Errors, TRW Software Series Report, TRW-SS-72-09, Redondo Beach, California, 1972.

35. Lipow, M., Maximum Likelihood Estimation of Parameters of a Software Time-To-Failure Distribution, TRW Systems Group Report, 2260.1.9-73B-15, Redondo Beach, California, 1973.

36. Littlewood, B. and Verrall, J.L., "A Bayesian Reliability Growth Model for Computer Software," Applied Statistics, Vol. 22, No. 3, 1973, pp. 332-346.

37. Littlewood, B., "A Reliability Model for Systems with Markov Structure," Applied Statistics, Vol. 24, No. 2, 1975, pp. 172-177.

38. Littlewood, B., "A Semi-Markov Model for Software Reliability with Failure Costs," Proceedings MRI Symposium on Software Engineering, New York, March 1976, pp. 281-300.

39. Maguire, B.A., Pearson, E.S. and Wynn, A.H.A., "The Time Intervals Between Industrial Accidents," Biometrika, Vol. 39, 1952, pp. 168-180.

40. Mann, N.R., Schafer, R.E. and Singpurwalla, N.D., Methods for Statistical Analysis of Reliability and Life Data, Wiley, 1974.

41. Miller, D.R., "Order Statistics, Poisson Processes and Repairable Systems," Journal of Applied Probability, Vol. 13, 1976, pp. 519-529.

42. Mills, H.D., On the Statistical Validation of Computer Programs, IBM Federal Systems Division, Gaithersbury, Maryland, Report 72-6015, 1972.

43. Miyamoto, I., "Software Reliability in On-Line Real Time Environment," Proceeding International Conference on Reliable Software, Los Angeles, California, April 1975, pp. 194-203.

44. Moeller, S.K., "The Rasch-Weibull Process," Scandanavian Journal of Statistics, Vol. 3, 1976, pp. 107-115.

45. Moranda, P.B., "Prediction of Software Reliability During Debugging," Proceedings Annual Reliability and Maintainability Symposium, Washington, D.C., January 1975, pp. 327-332.

46. Musa, J.D., "A Theory of Software Reliability and Its Application," IEEE Trans. on Software Engineering, Vol. SE-1, No. 3, September 1975, pp. 312-327.

47. Nelson, E.C., Software Reliability, TRW Software Series, TRW-SS-75-05, Redondo Beach, California, 1975.

48. Okumoto, K. and Goel, A.L., Classical and Bayesian Inference for the Software Imperfect Debugging Model, Syracuse University, Final Technical Report, RADC-TR-78-155, Vol. 2 (of 5), July 1978, AD A057-871.

49. Okumoto, K. and Goel, A.L., Availability Analysis of Software Systems Under Imperfect Maintenance, Syracuse University, Final Technical Report, RADC-TR-78-155, Vol. 3 (of 5), July 1978, AD A057-872.

50. Okumoto, K. and Goel, A.L., "Availability and Other Performance Measures of Software Systems Under Imperfect Maintenance," Proceedings Computer Software and Applications Conference, Chicago, Illinois, November 1978, pp. 66-71.

51. Proschan, F., "Theoretical Explanation of Observed Decreasing Failure Rate," Technometrics, Vol. 5, No. 3, August 1963, pp. 375-383.

52. Pyke, R., "Markov Renewal Processes: Definitions and Preliminary Properties," Annals of Mathematical Statistics, Vol. 32, 1961, pp. 1231-1242.

53. Rohatgi, V.K., An Introduction to Probability Theory and Mathematical Statistics, Wiley, 1976.

54. Ross, S.M., Applied Probability Models with Optimization Applications, Holden-Day, 1976.

55. Roussas, G.G., A First Course in Mathematical Statistics, Addison-Wesley, 1973.

56. Rye, P., et al., Software Systems Development: A CSDL Project History, The Charles Start Draper Laboratory, Inc., Final Technical Report, RADC-TR-77-213, June 1977, AD A042-186.

57. Schafer, R.E., et al., Validation of Software Reliability Models, Huges Aircraft Co., Final Technical Report, RADC-TR-78-147, June 1979, AD-A072-113.

58. Schick, G.J. and Wolverton, R.W., "Assessment of Software Reliability," 11th Annual Meeting of the German Operations Research Society, DGOR, Hamburg, Germany, September 1972; also in Proceedings Operations Research, Physica-Verlag, Wurzberg-Wien, 1973, pp. 395-422.

59. Schick, G.J. and Wolverton, R.W., "An Analysis of Computing Software Reliability Models," IEEE Trans. of Software Engineering, Vol. SE-4 No. 2, March 1978, pp. 104-120.

60. Schneidewind, N.F., "Analysis of Error Processes in Computer Software," Proceeding International Conference on Reliable Software, Los Angeles, California, April 1975, pp. 337-346.

61. Shooman, M.L., "Probabilistic Models for Software Reliability Prediction," Statistical Computer Performance Evaluation, W. Freiberger (Editor), Academic Press, 1972, pp. 485-502.

62. Snyder, D.L., Random Point Processes, Wiley, 1975.

63. Sukert, A., A Software Reliability Modelling Study, In-house Technical Report, RADC-TR-76-247, August 1976, AD A030-437.

64. Sukert, A.N., "An Investigation of Software Reliability Models," Proceeding Annual Reliability and Maintainability Symposium, Philadelphia, Pennsylvania, January 1977, pp. 478-484.

65. Sukert, A. and Goel, A.L., "Error Modelling Applications in Software Quality Assurance," Proceedings Software Quality and Assurance Workshop, San Diego, California, November 1978, pp. 33-38.

66. Sukert, A. and Goel, A.L., "A Guidebook for Software Reliability Assessment," Proceedings Annual Reliability and Maintainability Symposium, San Francisco, California, January 1980, pp. 188-190.

67. Thayer, T.A., Lipow, M. and Nelson, E.C., Software Reliability Study, TRW Defense & Space Systems Group, Final Technical Report, RADC-TR-76-238, August 1976, AD A030-798.

68. Trivedi, A.K. and Shooman, M., Computer Software Reliability: Many State Markov Modeling Techniques, Polytechnic Institute of New York, Interim Report, RADC-TR-75-169, July 1975, AD 014-824.

69. Wagoner, W.L., The Final Report on A Software Reliability Measurement Study, Technology Division, The Aerospace Corp., Report No. TOR-0074 (4112)-1, El Segundo, California, August 1973.

70. Willman, H.E., Jr., et al., Software Systems Reliability: A Raytheon Project History, Raytheon Co., Final Technical Report, RADC-TR-77-188, June 1977, A040992.

71. Yau, S.S. and MacGregor, T.E., On Software Reliability Modeling, Interim Report, Northwestern University, RADC-TR-79-129, June 1979, A072380.

APPENDIX A

DESCRIPTION OF SOFTWARE FAILURE MODELS

A.1 Stochastic Models for Times Between Software Failures

One of the earliest studies to develop a model for software reliability was undertaken by Jelinski and Moranda (Reference 31). They developed a model for the time between software failures, making the assumption of a uniform failure rate. In other words, the software error detection rate at any time is assumed to be proportional to the current error content (the number of remaining errors) of the tested program. It is also assumed that one error is removed/eliminated whenever a software failure occurs. If N is the initial number of errors in the system, the number of errors remaining after $(i-1)$ errors are removed will be $\{N-(i-1)\}$. If φ is the proportionality constant, the hazard rate or the error detection rate between the $(i-1)$st and the ith failures is

$$z(x_i) = \varphi[N-(i-1)] .$$

Then the Probability Density Function (pdf) of X_i, the time between the $(i-1)$st and the ith failures, is

$$f(x_i) = \varphi[N-(i-1)] \cdot e^{-\varphi[N-(i-1)]x_i} .$$

This constitutes the basic model of the so-called De-Eutrophication process. Statistical inference about the unknown parameters, N and φ, was discussed by Lipow (Reference 35) who obtained the maximum likelihood estimates and the variance-covariance matrix for N and φ. Forman and Singpurwalla (Reference 21) also proposed a method based on solving the difference equations in N and φ.

Moranda (Reference 45) modified the De-Eutrophication process by assuming the failure rate to decrease geometrically rather than decreasing in constant steps. He further incorporated the class of non-fatal errors, while the failure rates of successive errors form a geometric progression whose initial term is D and whose ratio is k. This is a superposition of a geometric De-Eutrophication process and a Poisson process with parameter θ. The model can now describe the burn-in phase by a De-Eutrophication process as well as the steady state by a Poisson process. For a combination Geometric De-Eutrophication and Poisson Model, the failure rate between the (i-1)st and the ith failures is given by

$$z(x_i) = k^{i-1} D + \theta .$$

Schick and Wolverton (Reference 58) developed a model whose hazard rate depends on the testing time as well as the number of remaining errors. They assumed that the hazard rate is a linear function of testing time, i.e.,

$$z(x_i) = \varphi[N-(i-1)]x_i$$

where N and φ represent the initial error content and a proportionality constant, respectively. It turns out that the distribution of the time between the (i-1)st and the ith failures is a Rayleigh distribution with parameter $\varphi[N-(i-1)]/2$. The estimates of N and φ can be obtained by the method of maximum likelihood.

Schick and Wolverton (Reference 59) postulated another model in which the hazard rate is a parabolic function, instead of a linear

function, of testing time x_i. The hazard function for this model is

$$z(x_i) = \varphi[N-(i-1)](-ax_i^2 + bx_i + c), \qquad a,b,c > 0 .$$

This yields an increasing number of errors while a debugging effort is in full force, then reaches a maximum, and finally declines as the number of remaining errors is drastically reduced.

A Bayesian approach was taken by Littlewood and Verrall (Reference 36) to develop a software reliability growth model. The underlying distribution of the time between the (i-1)st and the ith failures is an exponential distribution with rate λ; i.e.,

$$f(x_i|\lambda) = \lambda e^{-\lambda x_i} .$$

The failure rate λ is treated as a random variable with a Gamma distribution with shape parameter α and scale parameter $\psi(i)$, an increasing function of i. The function $\psi(i)$ is assumed to be known although it may differ from program to program. Assuming a uniform prior distribution of α, one can construct a data-dependent pdf of the time to next failure. Thus, the pdf of x_{n+1} for given n observations $x_1, x_2, \ldots, x_n$ is given by

$$f(x_{n+1}|x_1,x_2,\ldots,x_n) = \frac{n\gamma^n}{x_{n+1}+\psi(n)} \frac{1}{[\gamma + \log\{\frac{x_{n+1}+\psi(n)}{\psi(n)}\}]^{n+1}}$$

where

$$\gamma = \sum_{i=1}^{n} \log\{\frac{\psi(i)}{\psi(i)+1}\} .$$

It was shown that the reliability improves if $\psi(i)$ increases more rapidly with i than a linear function of i. A goodness-of-fit test was also presented and its use shown by choosing $\psi(i)$ to be a polynomial in i.

Most of the stochastic models treat the software system as a black-box. Littlewood (Reference 37) studied a model in which he incroporated the internal structure of the software system. He assumed that the software was composed of several sub-programs which worked in continuous time by Markov switching among themselves and that the failures occurred according to a Poisson process. The failures in the overall program were then shown to follow, asymptotically, a Poisson process whose failure rate can be computed from the failure rates of the individual structural components. By considering the distribution of the cost associated with failures, it was shown by Littlewood (Reference 38) that the distribution of the total vector cost due to failures of subprograms during $(0,t)$ is, asymptotically, multivariate normal.

A key assumption made in most of these models is that the errors are removed with certainty when detected. However, as pointed out in Miyamoto (Reference 43) and Thayer, et al. (Reference 67), in practice errors are not always corrected when detected. The above models do not provide an explicit solution for such situations.

To overcome this limitation, Goel and Okumoto (References 26, 28, 30) developed an Imperfect Debugging Model (IDM). In this model, the number of errors in the system at time t, $X(t)$,

is treated as a Markov process whose transition probabilities are governed by the probability of imperfect debugging. Times between the transitions of X(t) are taken to be exponentially distributed with rates dependent on the current error content of the system. Expressions are derived for performance measures such as the distribution of time to a completely debugged system, distribution of the number of remaining errors and software reliability. Okumoto and Goel (Reference 48) discussed methods for obtaining the maximum likelihood estimates and confidence regions for the parameters N (the initial error content), q (probability of imperfect debugging), and λ (failure rate per error).

The reliability models based on time between software failures are summarized in Table A.1.

TABLE A.1

A SUMMARY OF THE SOFTWARE RELIABILITY MODELS BASED ON TIME BETWEEN FAILURES

Model	Hazard Rate
De-Eutrophication Process (Jelinski & Moranda)	$z(x_i) = \varphi[N-(i-1)]$
Geometric De-Eutrophication Process (Moranda)	$z(x_i) = k^{i-1} D$
A Combination Geometric De-Eutrophication and Poisson Model (Moranda)	$z(x_i) = k^{i-1} D + \theta$
Linear De-Eutrophication Process (Schick & Wolverton)	$z(x_i) = \varphi[N-(i-1)]x_i$
Parabola De-Eutrophication Process (Schick & Wolverton)	$z(x_i) = \varphi[N-(i-1)]\{-ax_i^2 + bx_i + c]$
Data-Dependent Model (Littlewood & Verrall)	$z(x_{n+1}\|x_1,x_2,\ldots,x_n) = \frac{n}{x_{n+1}+\psi(n)} \frac{1}{\log \prod_{i=1}^{n} [\frac{\psi(i)+x_i}{\psi(i)}]}$
Imperfect Debugging Model (Goel and Okumoto)	$z(x_i) = [N-p(i-1)]\lambda$

A.2 Stochastic Models Based on Number of Failures

One of the earliest models in this category was proposed by Shooman (Reference 61). He analyzed the actual error data from functional tests after different debugging times, and used the history of these errors to specify the error detection rate function, $\rho(\tau)$. Hence, the total number of errors removed during τ months of debugging is taken to be

$$\epsilon(\tau) = \int_0^\tau \rho(x)dx .$$

If we assume that the total number of errors in the program, E_T, is constant, and that no new errors are added during debugging, then $\epsilon(\tau) \to E_T/I_T$ as $\tau \to \infty$, where I_T is the number of instructions in the program. Then the number of errors remaining at time τ can be expressed as

$$\epsilon_r(\tau) = E_T/I_T - \epsilon(\tau) .$$

Assuming that the software failures occur due to the occasional traversing of a portion of the program which has one or more errors, the hazard rate in an operational phase for software which has been debugged for τ months must be proportional to the number of errors remaining at time τ; i.e.,

$$z(t) = K\epsilon_r(\tau)$$

$$= K[E_T/I_T - \int_0^\tau \rho(x)dx]$$

where K is an arbitrary constant. Since $z(t)$ is constant over the operational time t, the reliability is simply

$$R(t) = e^{-[K \epsilon_r(\tau)]t}$$

and the mean time to failure is

$$MTTF = 1 / [K \epsilon_r(\tau)] .$$

A similar approach was taken by Musa (Reference 46) to develop an execution time model. If we denote the number of inherent errors in the program by N_0 and the net number of errors corrected during the execution time τ by $\eta(\tau)$, then the number of errors remaining at time τ is

$$M(\tau) = N_0 - \eta(\tau) .$$

He assumed that (i) the errors in the program are independent of each other and are distributed at any time with a constant average occurrence rate per instruction throughout the program, (ii) various types of instructions are reasonably well mixed, and (iii) the execution time between failures is large compared to average instruction execution time. The hazard rate of the errors is then given by

$$z(\tau) = Kf\eta(\tau)$$

$$= KfN_0 - Kf\eta(\tau)$$

where K is the error exposure ratio and f is the linear execution frequency. Furthermore, assuming that the error correction rate $\frac{d\eta(\tau)}{dt}$ is equal to the error exposure rate, $z(\tau)$, he obtained the hazard rate

$$z(\tau) = KfN_0 e^{-Kf\tau} .$$

Hence for execution time τ', projected from τ, the reliability is given by

$$R(\tau,\tau') = e^{-z(\tau)\tau'} .$$

This is the basic execution time model. The model was then generalized by introducing an error reduction factor, B, and a testing compression factor, C. The relationship between the execution time and calendar time was also investigated by incorporating the limitations on the availability of resources (failure identification personnel, failure correction personnel, and computer time).

Taking a different approach, Schneidewind (Reference 60) studied the number of errors detected during a time interval and the collection of error counts over a series of time intervals, by assuming that the failure process is a non-homogeneous Poisson process with an exponentially decaying intensity function

$$d(i) = \alpha e^{-\beta i} , \qquad \alpha,\beta > 0 , \quad i = 1,2,\ldots .$$

As an extension to his earlier models, Moranda (Reference 45) developed a geometric-Poisson model assuming that the number, N_i, of errors occurring in the ith interval is governed by a Poisson distribution with parameter λk^{i-1}.

A.3 Availability Models

As mentioned earlier, an operational software system is subject to random failures caused by software errors in the system. The maintenance/debugging activity is then undertaken whenever a software failure occurs. The system goes through a maintenance phase to remove the cause of failure and becomes operational as soon as the maintenance activity is over. Trivedi and Shooman (Reference 68) developed an availability model by considering the sequence of operational and maintenance (up and down) phases of the software system. The distribution of times in both states was taken to be exponential with a state dependent parameter.

A generalized model for the operational and maintenance phases was developed by Okumoto and Goel (References 49, 50). In this model, the time to remove an error is assumed to follow an exponential distribution with a rate dependent on the current error content of the system. The sequence of operational and maintenance states of the software system is formulated as a semi-Markov process and expressions are obtained for system availability and other performance measures. They also developed a nomogram to explore the trade-offs between the expected time to a specified number of remaining errors, which determines the software operational performance, and the manpower requirements to achieve the desired objective.

A.4 Combinatorial Models

A.4.1 Capture-recapture sampling

Mills (Reference 42) formulated the problem of estimating the number of errors in a program by using a technique called capture-recapture sampling. In this technique, a program containing an unknown number n_I of indigenous errors is deliberately 'modified' by seeding a set of known errors, n_s. These errors could be then discovered in successive tests, each of which is considered a trial. Then, the joint probability of finding X_I indigenous errors and X_s seeded errors is given by a hypergeometric distribution.

Lipow (Reference 34) modified this problem by taking into consideration the probability, q, of finding an error (of either kind) in any test of the software. Then, for N statistically independent tests the probability of finding X_I indigenous and X_s seeded errors is given by

$$P_N(X_I=x_I, X_s=x_s; q, n_I, n_s)$$

$$= \binom{N}{x_I+x_s} \cdot q^{x_I+x_s} \cdot (1-q)^{N-x_I-x_s} \frac{\binom{n_I}{x_I}\binom{n_s}{x_s}}{\binom{n_I+n_s}{x_I+x_s}} \quad .$$

The maximum likelihood estimates of q and n_I are given by

$$\hat{q} = \frac{x_I + X_s}{N}$$

and

$$\hat{n}_I = \begin{cases} \left[\frac{x_I}{x_s} \cdot n_s\right] & \text{if } x_I + x_s \geq 1 \\ 0 & \text{if } x_I + x_s = 0 \\ x_I n_s & \text{if } x_s = 0 . \end{cases}$$

Basin (Reference 5) suggested a somewhat different procedure, the so-called two-state edit procedure, where one programmer searches for defects and records n_1 errors out of a total of N unknown indigenous errors. A second programmer edits the program independently and finds, say, r errors. The two lists of errors are then compared. The probability that the same k errors are found by the two programmers is given by a hypergeometric distribution.

A.4.2 Input data domain considerations.

Software reliability assessment based on program structure has been proposed by Nelson (Reference 47). The reliability of a computer program here is defined as the probability of the program being correct on any given run. Data sets are used to execute the program structure. Each input data set proceeds through a sequence of segments, called a logical path, with a branch to a new segment taking place at the exit of each segment. The input data space, E, is partitioned into a small number of disjoint subsets, z_j, $j = 1,2,\ldots,k$, to produce the operational profile probability distribution, $P(z_j)$. If a program is executed a total of n times and f_j failures are observed out of n_j runs using points from z_j, then an estimate of program operational usage reliability is given by

$$\hat{R}_1 = 1 - \sum_j \frac{f_j}{n_j} P(z_j) .$$

Further, assuming that the test cases (i.e., n executions of the program) are identically proportional to $P(z_j)$, an estimate of the observed (or assessed) reliability is given by

$$\hat{R}_2 = 1 - \frac{1}{n} \sum_j f_j .$$

The techniques for developing a set of test cases which serve to accomplish a certain objective, e.g. assurance that each and every structural element would be exercised at least once during the execution of the program with the complete set of test cases, was discussed by Brown and Lipow (Reference 8). They used this technique to show its applicability on two fairly small programs.

A.5 Model Comparisons

Very few studies have been reported that compare the performance of various models. A comprehensive study for this objective was undertaken by Sukert (References 63, 64). In this study he analyzed the data from a large command and control system by using several software failure models. As a result of this study, he pointed out the limitations and difficulties in using these models. A limited comparison of the models from the quality assurance viewpoint has been reported by Sukert and Goel (Reference 65). A description and comparison of models has been given by Yau and MacGregor (Reference 71).

Schick and Wolverton (Reference 59) compared various models and indicated that the model they had developed (Reference 58), seems to perform better than others. Recently Angus, Schafer and Sukert (Reference 3) and Schafer, et al. (Reference 57) completed a comparative investigation of several models from the validation point of view. They analyzed several failure data sets and pointed out the difficulties of parametric estimation and other limitations of these models.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.